FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Unmanned Air Vehicle Based Localization and Range Estimation of Wi-Fi Nodes

Guilherme Carvalho

Master in Electrical and Computers Engineering

Supervisor: Professor João Tasso de Figueiredo Borges de Sousa

February 28, 2014

Unmanned Air Vehicle Based Localization and Range Estimation of Wi-Fi Nodes

Guilherme Carvalho

Master in Electrical and Computers Engineering

February 28, 2014

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Unmanned Air Vehicle Based Localization and Range Estimation of Wi-Fi Nodes"

foi aprovada em provas realizadas em 11-02-2014

o júri

Presidente Professor Doutor Mário Jorge Rodrigues de Sousa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Pedro Nicolau Faria da Fonseca
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Mestre João Tasso de Figueiredo Borges de Sousa
Assistente Convidado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Guilherme Gonçalves Coelho de Freitas Carvalho

Faculdade de Engenharia da Universidade do Porto

Abstract

Unmanned Aerial Vehicles (UAV) are of critical importance in the future of aerial surveillance and reconnaissance missions. Used to execute missions that are deemed dull, dirty or dangerous they enable us to save precious resources such as human life, time, and money. To safely substitute human interaction, the autonomous capabilities of these systems must be fully researched and developed to a degree that ensures that unmanned aviation can substitute its human counterpart, while ensuring critical safety standards.
This dissertation tackles the uncertainty of the communications range in regards to UAV missions. We propose to solve this problem by locating and estimating the range of the antennas used to establish a Wi-Fi Communications Network using the UAV as a mobile beacon in a localization system. We develop a plugin, dubbed *CommRanger* capable of integrating said functionalities into the current LSTS Command and Control Software: Neptus.
Our findings suggest that this is an advantageous system, as it is a cost efficient solution to implement, requiring no additional hardware or infrastructure, although using RSSI as a distance measurement presents some challenges, that will be addressed in the course of this document.

Resumo

Veículos Aéreos Não Tripulados (UAV) são de extrema importância no futuro de missões de reconhecimento aéreo. Usados para executar missões consideradas *dull, dirty or dangerous*, isto é, missões consideradas perigosas ou enfadonhas para serem executadas por operadores humanos. Para substituirem com segurança a interação humana, estes sistemas e as suas capacidades de autonomia devem ser estudadas e desenvolvidas até assegurarem que os requisitos de segurança são completamente cumpridos.
Esta dissertação aborda o problema da incerteza do alcance das comunicações, no contexto de missões UAV.
Durante esta tese propusemo-nos a resolver este problema localizando e estimando o alcance das antenas usadas para criar uma Rede de Comunicações Wi-Fi, usando o UAV como ponto de referência no sistema de localização. Desenvolvemos um plugin, de nome *CommRanger*, capaz de integrar estas funcionalidades integrado no software de comando e controlo usado pelo LSTS: Neptus.
Os nossos avanços concluem que é um sistema vantajoso, sendo uma solução de implementação eficiente a nível monetário, necessitando de nenhum *hardware* ou de infra-estruturas adicionais, embora o uso do *RSSI* como métrica de distância não se tenha apresentado sem os seus desafios, abordados no decurso do documento.

Agradecimentos

I would like to express my deepest gratitude to everyone involved in the creation of this dissertation.
Firstly, I would like to thank my supervisor Prof. João Sousa for the opportunity to work at the LSTS and and the precious advice and guidance in the realization of this dissertation.
I would also like to show my gratefulness to all the LSTS team for the warm welcome and for helping me a greater number of times than I would dare to count, in particular to Ricardo Silva for making the complex appear simple and the unfeasible, a done deal; to Daniel Silva, for always lending a helping hand, even when his own were full; and to Sérgio Ferreira for helping so many times, at any hour of the day or night, weekend or weekdays. Also, to the LSTS-AsasF team in particular for being, not only good co-workers, but also such good company during those long mission hours.
Many thanks to all my friends who put up with me during the ups and downs during the course of this dissertation, in particular to Fernando Barros for always being there for me.
I would like to demonstrate my deepest and most sincere thankfulness to my girlfriend Dulce for the enormous amounts of patience during the rougher patches of this dissertation and for the constant support throughout every stage of this venture.
Last but not least, I want to express my heartfelt gratitude for my family, and for their unconditional love and support.

Guilherme Carvalho

"It is better to fail in originality than to succeed in imitation."

- Herman Melville

Contents

List of Figures

List of Tables

Abbreviations and Symbols

UAV	Unmanned Aerial Vehicle
UAS	Unmanned Aerial System
LSTS	Laboratório de Sistemas e Tecnologias Subaquáticas
PITVANT	Projecto de Investigação e Tecnologia em Veículos Aéreos Não Tripulados
UP	Universidade do Porto
AFA	Academia da Força Aérea
RF	Radio Frequency
IMC	Inter-Module Communication
GPS	Global Positioning System
WGS-84	World Geodetic System 1984
DUNE	Unified Navigational Environment
GUI	Graphic User Interface
WLAN	Wireless Local Area Networks
LAN	Local Area Network
IEEE	Institute of Electrical and Electronics Engineers
AP	Access Point
RSSI	Received Signal Strength Indication
ISR	Intelligence, Surveillance and Reconnaissance
NLOS	Non-Line-Of-Sight
FSPM	Free Space Propagation Model
LOS	Line-of-Sight
ECEF	Earth-Centered Earth-Fixed

Chapter 1

Introduction

Since the dawn of flight, mankind has strived to conquer airspace not only as a means of transportation, but also as a way of providing a quick response to urgent situations.Flight is usually the preferred means of transportation for first response teams when dealing with several scenarios such as forest fires, earthquakes, shipwrecks among others. But, with great reward comes great responsibility and the risks associated with flying have always been tremendous, especially in these kinds of scenarios. Because of this, humanity has recently began to resort to the use of Unmanned Aerial Vehicles (UAV) as a way of removing the human element on tasks considered dangerous, dirty or dull. These missions can be performed either autonomously or semi-autonomously to provide a wide array of services and utilizations. Not only do they minimize the risk to human life, they are also a much cheaper solution and they are a lot faster to deploy since the preparation time is much shorter.

On certain applications using UAVs, we require a constant communication with the aircraft in order to extract information in real-time. Applications such as **Search and Rescue**, **Forest Fire Monitoring**, **Area Surveillance** and many more, require a constant stream of video and other data. This means that, whatever communication protocol is chosen, it must be able to maintain an open data link for the duration of the mission.

Communications with an Unmanned Aerial Systems (UAS) are necessarily wireless in nature, allowing the transmission of data between two points without any physical link between them. Wireless communications are nothing more than electromagnetic radio waves transmitting information from one point to another and as such are subject to a plethora of different interference sources that are impossible to predict, which in turn further complicates the task at hand.

1.1 Motivation

At *Laboratório de Sistemas e Tecnologias Subaquáticas* (LSTS), we aim to provide a various array of services using UAVs that strive to fulfill a myriad of different requirements proposed by each client. One of these requirements is real-time data streaming, be it video, image or text based.

In any situation where we have constraints related to the data link between aircraft and ground station, we are currently relying on basic estimations and little to no empirical evidence.
When planning to perform one of these missions, typically we consider aspects such as the UAV better suited for the service and in what area will the service be deployed. The area of the mission will dictate what sort of communication range will be required and as such, what type of antenna and how many antennas we will need.
Typically, a mission is successful when we can provide said service with no faults, conforming to all requisites. To ensure that the real-time data streaming requisite is satisfied, we must then make sure that the communication network is setup in such a way that completely covers the service area with a certain link quality guaranteeing a certain transmission rate, depending on the data being transferred. A study pertaining network deployment in UAV mission context is then recommended. If we were to give the operator a **map of the communications coverage**, integrated in the toolchain already used to command and control the vehicles, he would then be aware of the limitations of the current network allowing him to take action to improve it or reduce the operational area. In turn, this would allow us - the service provider - to quickly study and map the communications network, decreasing our network deployment time by reducing the amount of trial and error attempts, and guaranteeing that the mission requisites will be provided and safeguarded.
During these missions, the consequences of losing communications can be severe. Losing communications during an important mission like the ones stated above can signify the failure of the entire mission. What can then be done to minimize the chances of that happening? If the UAV's operator is provided information about the state of the communications network on the mission area, this would give him, not only a better grasp of where it is safe to operate, but also a much **higher awareness**, when deploying the wireless network, of where to deploy antennas in order to completely provide the necessary communications for a desired area.

1.2 Problem Statement

In this dissertation we tackle the uncertainty of the mobile system - the UAV - regarding its range of communications to the stationary ground station. We aim to provide operators with an estimate of the communications range available to their disposition during UAV missions or services, in order to increase their awareness and in turn, the efficiency with which they can perform the tasks at hand.
This meant creating a visual representation of the range and quality of a communications network in any given area. In pursuance of this goal, we've idealized the problem we face as:

- Locate the antennas used to establish the communication network in the mission area;
- Estimate their range for different data transmission rates;

In order to complete said project, we performed a comprehensive study regarding the systems used by the LSTS when deploying UAV services and afterwards dedicated our efforts in creating a program that directly integrated in their workflow that could be used whenever necessary.

1.3 Contributions

During this dissertation we created a program, dubbed *CommRanger*, that has the following capabilities:

- Provides the user with the location of the antennas used in the mission;
- Calculates not only the antenna's maximum communication range but also the several transmission rates achievable in relation to the distance;
- Can Save and Load computations done in order to be usable online and offline as a planning and studying tool;
- Integrates directly as a Plugin to the current LSTS Toolchain;
- Requires no additional hardware or infrastructures.

1.4 Scope

1.4.1 LSTS

Laboratório de Sistemas e Tecnologias Subaquáticas (LSTS) is an interdisciplinary research laboratory, established in 1997. Specialized in the design, construction and operation of unmanned underwater, surface and air vehicles and on the development of tools and technologies for deployment of networked vehicle systems.
This dissertation was developed using the vehicles and technologies built and developed in the LSTS and plans to integrate in the toolchain used in their developments, explained to greater detail in further chapters.

1.4.2 PITVANT Project

This thesis builds upon the *Projeto de Investigação e Tecnologia em Veículos Aéreos Não-Tripulados* (PITVANT) Project. Established in late 2008, the project results from the cooperation between *Academia da Força Aérea* (AFA) and *Universidade do Porto* (UP), and comprehends three separate phases, currently being in its third, and last, phase. The main goals now being:

1. Develop technologies in several areas, such as:
   (a) Project, build and test small and mid-size platforms;

   (b) Cooperative control of several vehicles with mixed initiative;

   (c) Develop systems interoperability;

   (d) Advanced vision systems;

   (e) Data fusion;

   (f) Navigation systems, applied to UAV systems;

2. Develop new operation concepts regarding small and mid-size UAV systems, applied in military operations and its subsequent validation in the field.

3. Testing of the systems and technologies developed in a broad spectrum of missions, ranging from military to civilian applications, featuring

   (a) *ISR* missions;

   (b) Combat missions enforced by cooperative UAV teams, some of them employing mixed initiative;

   (c) Evaluation tests featuring the new GNSS4-Galileo System;

4. Train staff to operate, maintain and define requisites for UAV systems;

1.5 Outline

We begin, in Chapter 2, by describing in greater detail the problem we aim to tackle in this dissertation by providing examples, describing the UAV system used by the LSTS and further expanding the problem.
Chaper 3 is dedicated to review the background theory necessary to the completion of this dissertation and in Chapter 4 we dedicated our efforts to researching and studying different approaches used to solve problems similar to our own.
In Chapter 5 we start developing the program we proposed to create, implementing each subsystem separately and finally fully integrating it into the LSTS toolchain. The experiments done to evaluate the system and the field tests performed during the course of dissertation are the subject of Chapter 6 and we end our dissertation with the concluding remarks and some recommendations for future work in Chapter 7.

Chapter 2

Stating the Problem

We've established that having some insight regarding the area of communication of an UAV can be of inestimable value to the operators, enabling them to increase their mission efficiency and increasing their foresight regarding mission safety. In order to achieve such a goal, one could simply estimate the range of an antenna by hand, and then constrain the flight to the calculated zone, but this solution would allow anyone to perform that task automatically, bridging the gap between development and commercial usage.
As we've stated previously, on certain missions, it is of vital importance to maintain a data-link between UAVs and the command center. As with all radio signals, communication range is tricky to determine and typically it is a barrier that isn't directly tackled in systems such as this one due to the fickle nature of electromagnetic signals. Usually, the operator has an idea of the maximum range based on empirical knowledge and works around it by trial and error.

2.1 An Illustrating Example

To better convey our idea, here is a brief practical example:
A client wants to buy a fleet of small, light and cheap UAV to use in the detection and monitoring of Forest Fires as described in (Casbeer et al., 2005) on which every UAV will have a certain area of coverage in order to detect fire and its propagation thanks to an onboard infra-red imaging camera. As posed by Casbeer et al., "(...)UAVs are also assumed to have limited communication range, which means they cannot upload data to the base station unless they are within range of the station(...)". Finding out what this range is would lead to a more efficient area coverage. Deploying our program would be done in two stages:

- Analyze the existing communications network by locating the present antennas and estimating their range using the plugin we developed;
- Relocate, remove or add antennas in order to create a more desirable communication network coverage.

In this particular example this would allow us to not only ensure that the communication between UAVs and the base station is guaranteed but also to better establish each UAV area of interaction. This is a perfect example of why the study and positioning of communications network is very important to some UAV mission scenarios and is worthy of more investigation.

2.2 System Description

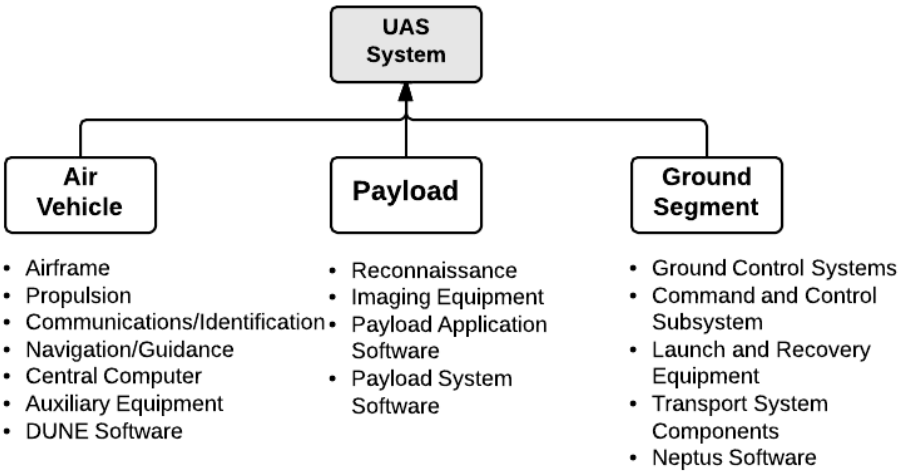


Figure 2.1: System Breakdown Structure

We will be integrating our developments in the system currently used by the LSTS in regards to UAS. This system is composed by three level 1 items: Air Vehicle, Payload and Ground Segment. For the purposes of this dissertation, we will focus solely on the Air Vehicle and Ground Segment as the Payload is irrelevant to our program.

2.2.1 Air Vehicle

Figure 2.2: LSTS X8 Air Vehicle

The platform used for the purposes of this dissertation was the Skywalker X-8 Flying Wing, assembled and modified for autonomous navigation in the LSTS. The X-8 is the ideal platform for fast algorithm testing, terrain mapping and operational surveillance due to its ease of deployment and quick recovery. It is controllable by a single laptop and is equipped for 5.0GHz Wi-Fi communication.
Two antennas are connected to the Wi-Fi radio, a Rocket M5 by Ubiquiti Networks. These antennas are placed in a 90º angle disposition, as shown in the following figure:

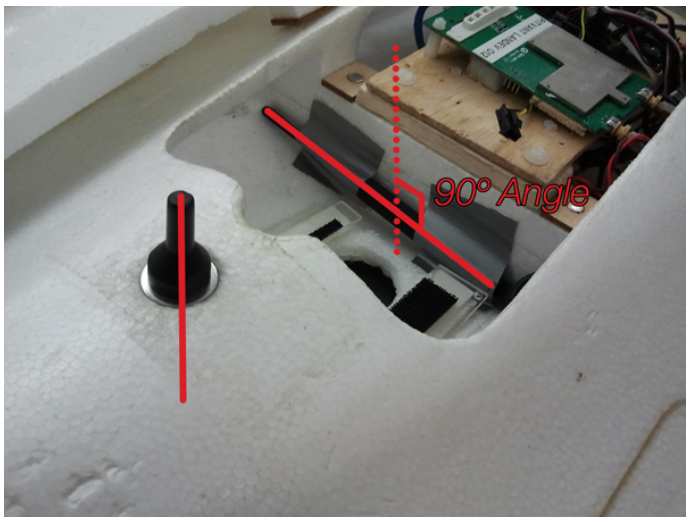


Figure 2.3: X8 Platform Antenna Disposition

2.2.2 Ground Segment

The Ground Segment is composed by several subsystems, depicted in Figure 2.1. For the intents of this dissertation, our Ground Segment was composed by:

- Manta Communications Gateway - a hub with capabilities to deploy 2,4GHz Wi-Fi and 5GHz Wi-Fi;
- Nanostation M: Used to establish the 5GHz Wi-Fi Communications Network, used to communicate with the UAS;
- Bullet M: Used to establish the 2.4GHz Wi-Fi Communications Network, used to communicate with the base station;
- Base Station: One or more Computers running Neptus;
- Launch System for the X8 Platform;

The Nanostation M is a powerful radio and high-gain antenna (up to 16dBi) combination that is capable of long distance communications with a 60^{o} angle of coverage vertically, as seen in the data sheet (nan, 2014). The Bullet M is a wireless radio with an Integrated Type N RF connector that can be connected directly to any antenna to create a powerful outdoor access point or bridge. We used it connected to a 2.4GHz omnidirectional antenna.
Both of these radio/antenna combinations are connected and powered via Ethernet to the Manta Communications Gateway, serving as a battery-powered communications hub.

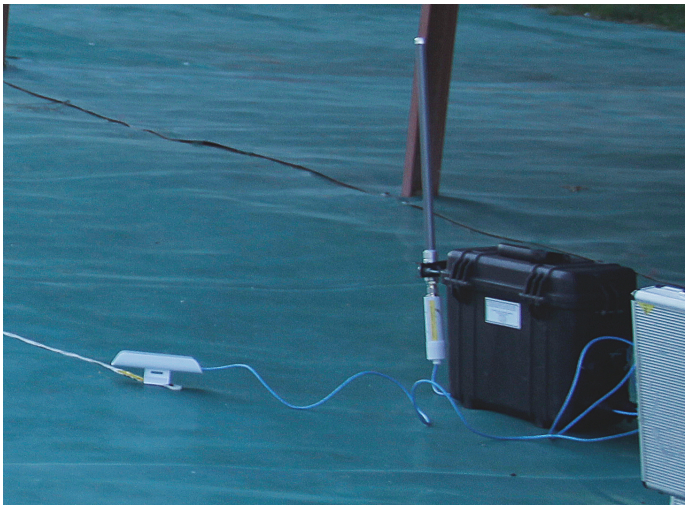


Figure 2.4: Manta Communications Gateway In Mission Scenario

In Figure 2.4 we can see both the Nanosation M (on the left) and the Bullet M (on the right) connected to the Manta, with communications network deployed for UAV communications and

for base station communications. The following figure illustrates a simplified diagram enacting communications in a mission scenario:

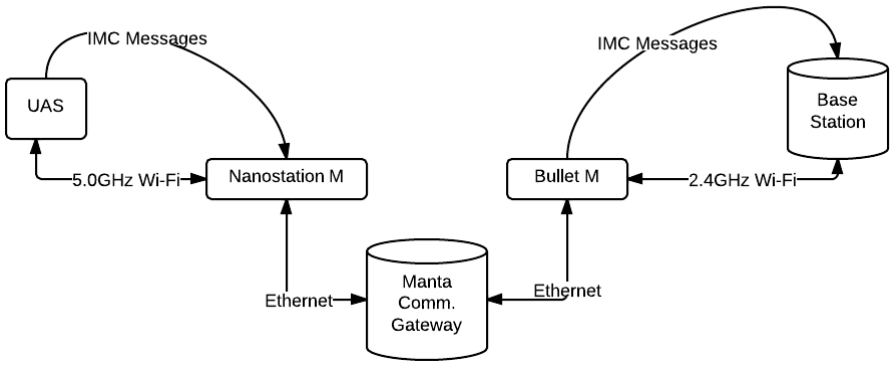


Figure 2.5: Deployment of Communications in a Mission Scenario

2.2.3 LSTS Toolchain: Onboard and Off-board Software

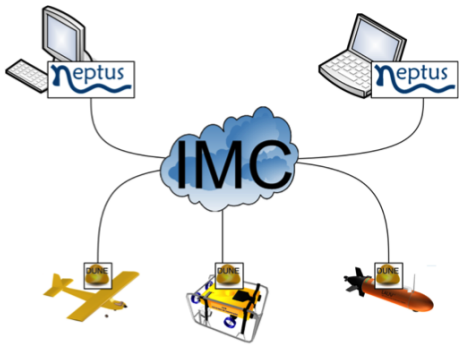


Figure 2.6: LSTS Toolchain: Neptus, IMC and Dune

The current LSTS Development toolchain consists of three in-house developed programs, each with their specific purpose: Neptus, IMC and DUNE.

Neptus is a Java based *C4I (Command, Control, Communication, Computer and Information)* mixed initiative (human operators in the control loop) framework for the coordination and control of various autonomous and semi-autonomous vehicles currently being developed by the *LSTS*. Neptus is composed of mission and vehicle planning, supervision and post-mission analysis modules(Dias et al., 2008). The framework was developed having adaptability and flexibility as top priorities to encompass the needs of several different systems, scenarios and operators. As such, it can be customized according to the individual needs of the operator and/or the mission.
Neptus allows the integration of independently developed plugins as compiled **.jar* files and, thus,

can be extended with new components. A plugin is a software component that adds a specific feature to an existing software application.

Information regarding the vehicle state, position, sensors, etc. is relayed to the operator via a base message protocol, the Inter-Module Communication (IMC).
IMC, or Inter-Module Communication, is a message oriented protocol designed and implemented by the LSTS to build interconnected systems of vehicles, sensors and human operators by exchanging real-time information via a shared set of messages that can be serialised and transferred over different means.(Martins et al., 2009)
There are several message groups comprised in the IMC Protocol, providing different layers for control and sensing. The layers we will be studying for the purposes of this dissertation are:

- Sensor Messages - Report sensor readings, by their respective hardware controllers. We will be particularly interested in the *GPS* sensor measurement, represented by the *GPS Fix* message. This message's payload contains the *WGS-84 Latitude, Longitude* and *Height* coordinates(LSTS, 2013).

- Navigation Messages - Report the vehicle's navigation state via the *Estimated State* message. This message broadcasts various information regarding the location, speed and rotation of the vehicle, thoroughly describing the momentary state(LSTS, 2013).

IMC abstracts hardware and communication heterogeneity by providing a shared set of messages that can be serialized and transferred between different means. It is also the mean used to convey messages from the hardware via DUNE.

DUNE: Unified Navigational Environment is the runtime environment for the software onboard the vehicle. It is used to write software that interfaces directly with the vehicle: code or control, navigation, communication, sensor and actuator access, etc. It is an operating system and architecture independent platform written in C++.

2.3 Problem Decomposition

The main goal of this dissertation is to provide both experienced and unexperienced users with an estimate of the usable communication range in a mission scenario.
After understanding the system we are interfacing with, we can now trace a path to the solution of the problem by analyzing it to a greater extent and further decomposing it into small problems we can tackle individually and later integrate into a fully functioning solution.
To know more about the state of the communications network - a 5.0GHz Wi-Fi Network - in a designated area is to estimate the range of each antenna deployed in that area. To do so, we must gather knowledge regarding two distinct subjects: Antenna Position, and said Antenna's Communications Range. We then consider the desired outputs of our system to be the Antenna

Position in global coordinates $AP(lat, lon, height)$ and associated Communications Range Value $ran(meters)$. Directly, we can describe our problem as a block diagram, with a set of points with global coordinates $P(lat, lon, height)$ and associated RSSI Values $R(rssi)$ being the inputs to our system:

$$P = \{p_1, p_2, p_3, ..., p_i\}, i \, \varepsilon \, \mathbb{N} \tag{2.1}$$

$$R = \{r_{p_1}, r_{p_2}, r_{p_3}, .., r_{p_i}\}, i \, \varepsilon \, \mathbb{N} \tag{2.2}$$

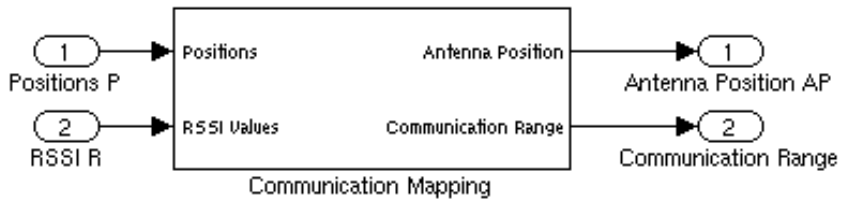


Figure 2.7: Simplified System Block Diagram

This is the program that will finally be integrated onto the current LSTS Toolchain - Neptus-IMC-Dune - keeping in mind the requirements of each system. Our solution needs to be highly flexible, easy to understand and available to anyone that operates a UAV in the LSTS and as such, keeping it simple and automated is a top priority.
After understanding the basic function of our program, we can now expand the Communication Mapping system and start to analyze each separate subsystem in order to create a subset of problems to approach individually.

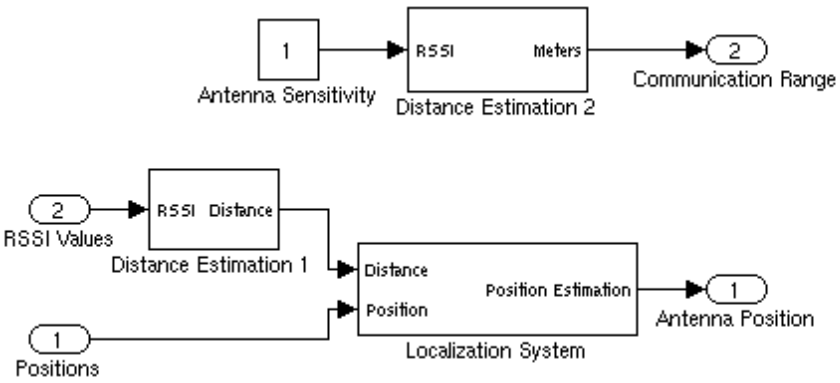


Figure 2.8: Expanded System Block Diagram

We will dedicate a subsection to each output, exploring the possibilities and problems we faced during the inception stage of this dissertation. In addition to those subsections, we will have another section dedicated to the implementation of the program in the LSTS toolchain.

2.3.1 Communications Range

Calculating the communications range of an Antenna is a fairly straightforward problem. Most antenna manufacturers provide **Sensitivity Values** for the connection established by an antenna. The sensitivity of an electronic device such as an antenna is the **minimum magnitude of input signal required to produce a specified output signal**. In the case of a radio antenna, it is expressed in *dBm*, which is an abbreviation for the power ratio in decibels of the measured power referenced to one milliwatt. Typically, an antenna data sheet provides **several sensitivity values for several maximum transmission rates** with an associated tolerance value.
The problem becomes simply a matter of, as shown in Figure 2.8, convert said sensitivity values, constant for any given antenna, to a distance value. Converting radio signals to distance based on transmitted power is a widely discussed and studied subject and thus **our problem is a matter of choosing an already developed signal propagation model and implement it in algorithm form.**
In pursuance of our goal, we must then choose a model that will take into account our test conditions and our variables:

- Free-Space - The algorithm will always be used in outdoor environments, therefore we can assume that there are no obstructions in most of the situations.
- Line-Of-Sight - Related to the previous point, typically, the airplane will maintain line of sight with the antennas at all times, since there are no obstructions in between them.
- High Gain Antennas - Since high gain antennas are used in our communications, we need to choose a model that can assume a gain different than unity.

There are several models that fit our requirements, and they will be disclosed and further discussed in Section 3.3.2

2.3.2 Antenna Position

To know the position of the antenna, we face the widely approached and studied problem of localizing a point in time and space. Generally, localization systems work by using three points with known positions to locate a fourth point with unknown position via localization algorithms such as **trilateration or triangulation**.

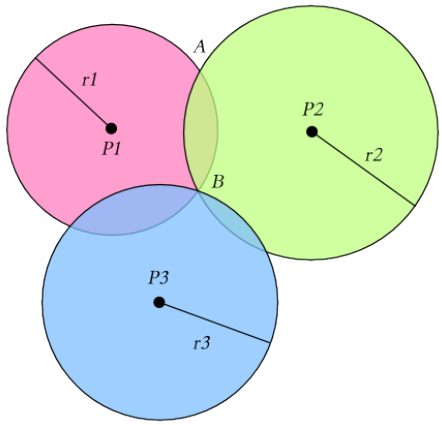


Figure 2.9: Localization Example

Shown in the figure above, given the set of reference points $P1, P2$ and $P3$ and their distances to a target node B, we can define a circular shape with the reference point as the center of a circle and the distance as its radius. The solution is then the point that satisfies the following condition, with $P1 = (x_1, y_1)$ and its distance to target node being d_1, and so forth, with n being the number of reference nodes:

$$(x+x_i)^2 + (y+y_i)^2 = d_i^2, i = 1,2,3,...,n \tag{2.3}$$

The main difference in our system is that we **do not have three points with known location** to compute a normal localization system. However, we do have an UAV. This means that, for stationary objects, **the UAV can simulate infinite reference points by acquiring distance estimation values throughout its trajectory** as presented in Figure 2.10 using the same nomenclature as in the previous example.

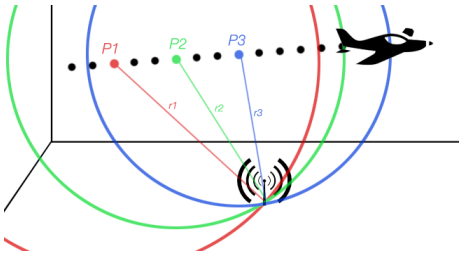


Figure 2.10: Localization Example using an UAV as Reference Node

As shown in Figure 2.8, we can define the reference points chosen as one of the inputs of our Localization System. The other input, however, needs to be a distance in meters. The only added hurdle is the fact that **our input is not a distance value, but a RSSI value**. RSSI values are expressed in *dBm*, like sensitivity values and thus, to overcome this drawback, **we must bring into play the same algorithm used in the conversion of sensitivity to distance**. Having as the input of that subsystem the RSSI values regarding one antenna will yield a distance estimation to that same antenna.
Selecting these reference points is far from trivial and will determine the success and mean error of the Localization System. We must take into consideration these factors when deciding the optimal points to input:

- Collinearity and Coplanarity - In order to achieve a localization, our reference nodes must not be collinear, when working two-dimensionally or coplanar, working tri-dimensionally. So, a decisive factor in the success of our localization is guaranteeing this condition is met.
- Aircraft Attitude - Certainly we will face some unforeseen effects of the aircraft's attitude in our measurements. Be it speed, roll, direction, we will need to test which situations are better suited to use as reference nodes.
- Distance Estimation Error - We need to choose the nodes on which the algorithm discussed in the previous section has less error. There are several influencing aspects regarding the use of signal models, and we need to take those into account.

We can clearly see that implementing an decision mechanism that chooses which reference nodes to use as inputs to our system in order to minimize the localization error is a decisive factor, and will play a large role in the success of the whole system. The algorithms chosen will also be discussed in greater detail in Section 3.4

2.4 Software Implementation

All this would be meaningless if the operator didn't have access to this information in a streamlined and easy to understand process. The Graphic User Interface *(GUI)* used for the mission planning and control is a software, developed by LSTS, named *Neptus*.

Developed in order to command and control fleets of unmanned vehicles, it allows operators to observe real-time data of networked vehicles and to revise data from previous missions. This makes it the perfect framework on which to develop our plugin which will be loaded into the toolbar represented in Figure 2.11.

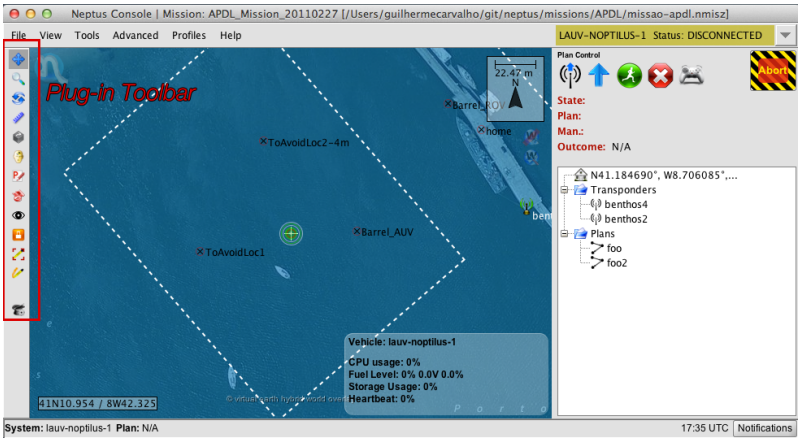


Figure 2.11: Neptus Mission Console

The implementation of this Plug-In will be made in the Java programming language, and integrated into the current build-path of the Neptus software in order for it to be available to any UAS operator when loading the program. The dimension of the programming work that needs to be accomplished is as follows:

- Implement a distance estimation algorithm;
- Implement several solutions of Localization Systems, from position computation algorithms, to the localization algorithm that brings all the systems together;
- Integrate them into a Plugin in the Neptus framework;
- Design and implement the graphical aspect of the program

2.5 Summary

This chapter focused mainly on stating the problem at hand and thoroughly clarifying the dimension and application of this dissertation. We started by providing an illustrative example of where our solution would be of use, and describing the system we will integrate this program in. After defining those two sections, the remaining sections were dedicated to decomposing and breaking down the larger problem into several smaller problems that can be addressed individually and posteriorly integrated into one final solution.

Chapter 3

Background Material

This chapter is dedicated to cover all the underlying concepts and technologies necessary to tackle the problem presented by this dissertation. We will dedicate a section to each important concept approached in the previous chapters in order to inspire us to devise an implementation for the problem approached.

3.1 Unmanned Aerial Vehicles

UAVs are uninhabited, reusable motorized aerial vehicles which can be remotely controlled, semi-autonomous, autonomous, or any combination of these capabilities and can carry several types of payload, giving them the abilities to perform specific missions within the earth's atmosphere or beyond. They are split into several categories, by weight, flight altitude and speed as shown in the following table:

UAV Category	Maximum Gross Takeoff Weight (lbs)	Normal Operating Altitude (ft)	Speed (km/h)	Representative UAS
Group 1	0-20	< 1,200 AGL	100	*Wasp III, Pointer*
Group 2	21-55	< 3,500 AGL	< 250	*SilverFox, ScanEagle*
Group 3	< 1,320	< 18,000 MSL	< 250	*RQ-7B*
Group 4	> 1,320	< 18,000 MSL	Any Airspeed	*MQ-5B, MQ-8B*
Group 5	> 1,320	> 18,000 MSL	Any Airspeed	*MQ-9A, RQ-4 GlobalHawk*

Table 3.1: UAS categories (Borges Sousa et al., p. 7)

For the purposes of this project, Group 1 UAVs are the ones being considered, for they are the most relevant in the PITVANT project.

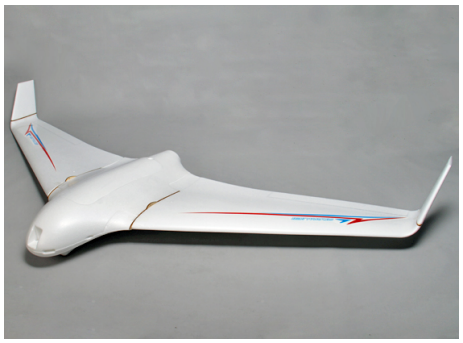


Figure 3.1: Group 1 UAV used in the PITVANT Project

The scope of UAV applications stretches from military and humanitarian missions, for instance, mine detection, perimetric surveillance and helicopter flight path reconnaissance, to civilian applications, such as crop spraying, fertilizing and seeding purposes. (van Blyenburgh, 1999) Since the dawn of flight, there has been the concern for human life, and safety. Therefore, on certain critical missions and endeavors, there was a quest to develop a system that didn't involve risking the well-being of the pilots. Along with that responsibility, on some missions, the expenditure of human resources was considered wasteful due to the repetitive nature of the processes involved. Thus, Unmanned Aerial Systems were created, to engage missions considered long and tiring for aircraft pilots, or that would present a high risk factor.(Haulman, 2003).

There are several advantages to using UAVs or Manned Reconnaissance Aircraft, such as:

Table 3.2: UAV and Manned Reconnaissance Aircraft Advantages (Haulman, 2003, p. 2)

Unmanned Aerial Vehicles	Manned Aircraft
No casualties	Faster
Less expensive per aircraft (the cost of the original Predator was a fifth that of an F-16)	Direct control and more situational awareness allows greater flexibility
Can fly longer missions to provide near real-time reconnaissance (not subject to human endurance limitations)	Better performance in bad weather
Reduces time between target identification and destruction	Airframes incorporate more stealth technology
Space and payload for pilot and life support equipment available for other uses	Not as dependent on ground and satellite signals that may fail
Can fly into more hostile environments	More likely to return if hit
Smaller (more difficult to detect than manned aircraft without stealth)	Refuelable by aerial tanker
Easier to store and ship	Tolerates rougher runways

There has been a growing interest in small UAVs like the one in Fig 3.1, for they are considered expendable, thanks to their cheap airframes, and relatively simple assembly. In spite of their expendability, in regards to Auto-Pilot implementation, there is still the need to develop a robust and accurate system in order for the tasks to be completed with efficiency. As such, it is easy to understand the growth of the UAV programs in the scientific world, emphasizing the development of fully autonomous UAV systems, with rational decision making capabilities such as the ones being developed in the PITVANT Project.

3.2 Wi-Fi: An Overview

Wi-Fi is a commonly used technology that allows two or more electronic devices to exchange data using radio waves, defined as any WLAN (Wireless Local Area Network) products based on the IEEE's (Institute of Electrical and Electronics Engineers) 802.11 standards. Wireless Local Area Networks (WLANs) have become one of the most popular means of connecting our equipment. Most public places now offer Wi-Fi Connections, and at home, it gives us the benefits of reduced cabling, ease-of-use, and network-based remote control.
Typically, a Wi-Fi system consists of three parts: a network line; an Access Point (AP) and one or more WLAN adapters. The network line is connected to the AP, who acts as a transmitter-receiver pair and establishes the wireless link between the network line and the WLAN adapters which are installed in the devices wishing to communicate via Wi-Fi.
Wi-Fi is specified and governed by the IEEE 802.11 standards and utilises 2.4GHz or 5GHz bands to communicate.The current, most used IEEE standard is the 802.11g and 802.11n, providing data rates of up to 54Mbit/s and 600Mbit/s respectively.
When interfacing with an electronic device connected to a Wi-Fi Network, the strength of the signal is given by the RSSI (Received Signal Strength Indicator) value, expressed in units of decibels with respect to milliwatts (dBm).

3.3 Signal Propagation Models

Received Signal Strength Indication (RSSI) is a measure, in dB of the power in a received radio signal. Since the signal is nothing more than an electromagnetic wave, the general concepts of electromagnetic wave propagation apply, and studying those, we can further understand the way this radio signal traverses the medium.

3.3.1 Electromagnetic Waves: General Concepts

The study of Electromagnetic Waves and its behaviour is very complex and could itself be the subject of a master's thesis. Nevertheless, in order to achieve our goal, we must attempt to understand,

if only superficially, what influences the way in which these waves traverse the environment. There are several mechanisms that affect the radio wave transmission:

Reflection

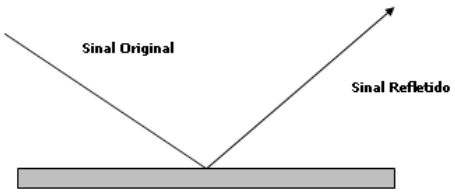


Figure 3.2: Signal Reflection, extracted from (Battisti, 2013)

In the same fashion as the everyday reflection phenomenon such as light hitting a mirror, signal waves can also be reflected (Figure 3.2). This happens when a signal wave encounters a object larger than it's wavelength in its path. Typically, the intensity of the reflected wave (which depends on the conductivity, permeability and permittivity of the object) is lower than the original wave due to absorption or as a result of some of the signal passing into the medium.

Reflection can cause several problems in RF Communications such as degradation of the original signal, or faults in the intended coverage area. It can also cause *Multipath Propagation*, a phenomenon that occurs when the signal reaches the receiving station by two or more paths, which could cause interference and phase shifting.

Refraction

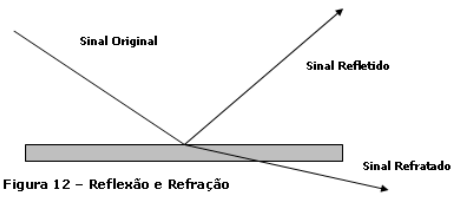


Figure 3.3: Signal Refraction, extracted from (Battisti, 2013)

Similarly to the previous mechanism, waves can also be Refracted (Figure 3.3). Different mediums have different refractive indexes and it is found that the direction of the electromagnetic wave changes as it passes from one refractive index to another. In long-distance communications, as the signal passes through several areas with different refractive indexes and its direction gets changed, it may never reach its destination or face the same *Multipath Propagation* that was discussed above.

Diffraction

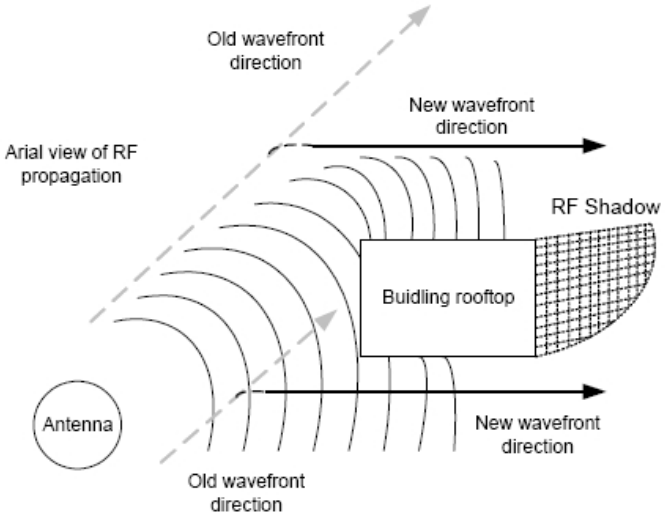


Figure 3.4: Signal Diffraction, extracted from (Battisti, 2013)

Diffraction (Figure 3.4) occurs when the signal is obstructed by an obstacle with irregular sized edges. This mechanism allows electromagnetic waves to go around obstacles, creating shadow regions between the emitter and the receiver. This means that, even if there is a large object in the transmission path, the receiver might still be able to maintain communication with the transmitter due to the Diffraction phenomenon.

Dispersion

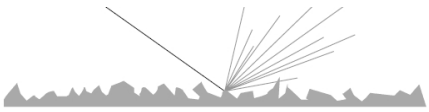


Figure 3.5: Signal Dispersion, extracted from (Battisti, 2013)

Dispersion (Figure 3.5) is what's experienced when the signal hits a rough surface, with irregularities of approximately the same size as the signal wavelength. The signal is then reflected in multiple directions simultaneously with a pronounced loss in amplitude resulting in a substantial degradation of the signal or even a complete signal loss.

3.3.2 Propagation Models

3.3.2.1 Linear Interpolation

Since the environments are significantly different from place to place, the simplest way to find the relationship between RSSI and Distance is to collect signal strength data from several known positions.
There are two types of errors coupled with this procedure. Signal Propagation Model errors, where the model derived isn't realistic enough, and NLOS (Non-Line-Of-Sight) errors, where the path between APs is partially obstructed by obstacles in the *Fresnel Zone*. The impact of these errors, and means to overcome them will be addressed in future sections. (Li et al., 2008)
Studies regarding the efficiency of RSSI as a candidate for Localization experiments show that other options might be less error-prone. RSSI, measure in decibels, has an inverse linear relation with distance, and can be plotted as seen in Fig 3.6:

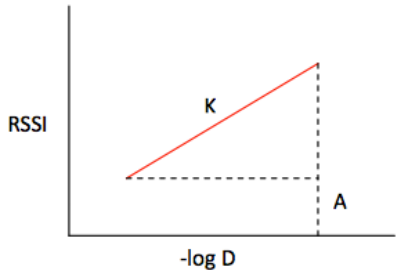


Figure 3.6: Expected relationship between RSSI and [-log (distance)].

Where K is the slope of the standard plot, and A is a constant value, and both these constants can be estimated through linear regression analysis on data points used to generate the standard curve. The following formulae would then be extracted:

$$RSSI = -KlogD + A \tag{3.1}$$

$$D = 10^{[(A-RSSI)/K]} \tag{3.2}$$

3.3.2.2 Free-Space Propagation Model

In situations where the receiver and the transmitter have a clear, unobstructed Line-of-Sight (LOS) Path between them, we can use the Free-Space Propagation Model (FSPM) as a mean to predict received signal strength.(Rappaport et al., 1996)
The FSPM predicts that the received power decays as a function of the separation between transmitter-receiver pairs, given by the Friis' Free Space equation:

$$P_r(d) = \frac{P_t * G_t * G_r * \lambda^2}{(4\pi)^2 * d^2 * L} \tag{3.3}$$

P_t - Transmitted Power

$P_r(d)$ - Received Power

G_t - Transmitter Antenna Gain

G_r - Receiver Antenna Gain

d - Transmitter-Receiver Separation in meters *(m)*

L - System Loss Factor $L \geq 1$

λ - Wavelength in meters *(m)*

We can see in Equation 3.3 that the received power falls off as the square of the distance, thus implying that the power decays with distance at a rate of 20 $dB/decade$.
The *path loss*, representing signal attenuation in dB, is the difference between the transmitter and receiver effective power and may include the antenna gains. The path loss for the model stated above, representing signal attenuation, is given by:

$$PL(dB) = 10log\frac{P_t}{P_r} = -10log\left[\frac{G_t * G_r * \lambda^2}{(4\pi)^2 * d^2}\right] \tag{3.4}$$

This model is only a valid predictor for values of d in the far-field of the transmitting antenna, also called the *Fraunhofer region*, which is given by:

$$d_f = \frac{2D^2}{\lambda} \tag{3.5}$$

where D is the largest physical linear dimension of the antenna.Typically, the reference distances for systems operating in the $2GHz$ range are $1m$ for indoor environments and $100m$ to $1Km$ for outdoor environments. (Rappaport et al., 1996)

These formulas can help determine the distance based only on the RSSI values measured, but they are a major simplification, not taking into account several influencing effects such as the ones discussed in Section 3.3.1 regarding the propagation of the signal itself, and others regarding the hardware used to measure and read the RSSI values.(Blumenthal et al., 2007)

3.3.2.3 Ground Reflection (2-Ray) Model

As we know, when discussing wireless communications, a single direct path between transmitter and receiver is frequently unattainable. As such, the FSPM Equation (3.3) is most of the times inaccurate when used alone. The Ground Reflection Model considers both the direct path and a reflection from the ground to estimate the Path Loss. This method has been found reasonably accurate predicting distances for mobile radio system using tall towers (heights above $50m$) as well as for LOS microcell channels in urban environments, as seen in (Rappaport et al., 1996, p. 86).

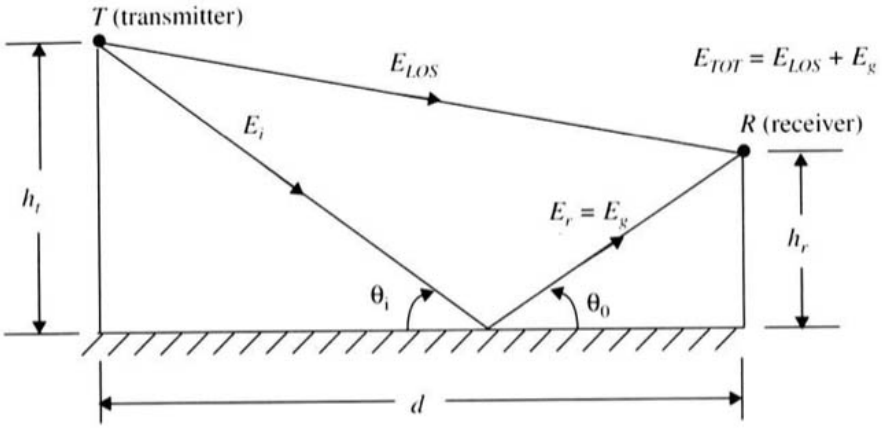


Figure 3.7: Ground Reflection or 2-Ray Model (Figure from (Rappaport et al., 1996, p.86))

We can see, in the figure represented above, that the total received $E-field$, E_{tot} is the result of the direct LOS component, E_{LOS} and the ground reflected component, E_g. So, in short, the total power seen by the receiver is given by:

$$P_r(d) = P_t * G_t * G_r * \frac{h_t^2 * h_r^2}{d^4} \tag{3.6}$$

And, subsequently, the *path loss* is extracted from the following equation:

$$PL(dB) = 40log(d) - [10log(G_t) + 10log(G_r) + 20log(h_t) + 20log(h_r)] \tag{3.7}$$

We can see in Equation 3.6 that the received power falls off with the distance raised to the fourth power, which translates to a rate of 40 $dB/decade$. Compared to the FSPM Equation (3.3), this models presents a much more rapid path loss. (Rappaport et al., 1996, p 89)

3.4 Localization Systems

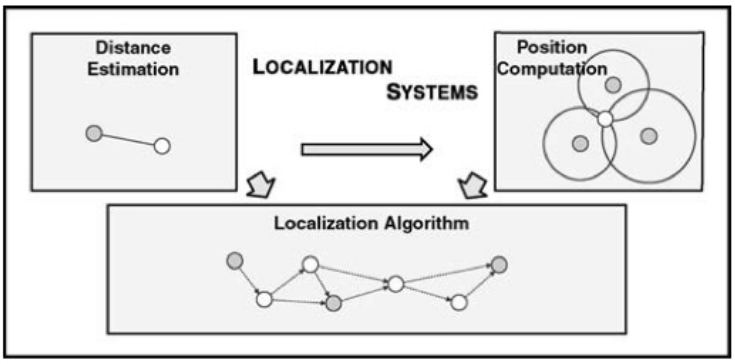


Figure 3.8: Localization System - Figure from (Boukerche, 2008))

As seen in Figure 3.8, a typical Localization System has three distinct components:

Distance Estimation - This component is responsible for the estimation of the distance and/or angles between two nodes. This information is later fed into the localization algorithm.

Position Computation - Here, the position of the node is calculated based on available information of the distances and position of the reference nodes.

Localization Algorithm - The main component of the localization system. It determines how the information will be handled in a way that every node, or most of them achieve a successful position estimation.

3.4.1 Distance Estimation

Various methods can be used to estimate the distance. They have different precision, but different costs and resource requirements and not all of them are readily available on every sensor. For the purposes of this thesis, since we aimed to integrate the localization system with the already available means, we used RSSI as the Distance Estimator.
This method derives the distance between the two nodes based on the strength of the signal received by one of the nodes. It has the advantage of requiring no additional hardware, but we must be wary of noise and interferences, as it's very sensitive and can produce large inaccuracies. This will be further developed and explained in Section 3.3.
The danger of channel noise and interferences can result in one or several of the following situations:

Uncertainty - There can be a situation where the various spheres do not intersect at a single location. This can happen under noisy ranging measurements, and will result in a failed attempt to locate a single point;

Nonconsistency - A single node can have many reference neighbours. Any subgroup of them can locate this node by multilateration. The computed result can vary if different groups of references are chosen thus resulting in a non consistency error;

Ambiguity - The flip ambiguity, where a reference creates a mirror through which the position can be reflected, occurs very often under noisy ranging measurements or under poorly connected networks;

Error Propagation - The errors from each step of multilateration can propagate and accumulate thus resulting in a larger error.

3.4.2 Position Computation

When a target node, the node whose position we are trying to compute, acquires sufficient information about distance related to the reference nodes, it can compute its own position using one of

the methods described in this section.
Several methods can be used calculate a node's position, such as Trilateration, Multilateration, Triangulation, Probabilistic approaches, Bounding Boxes and several others. The choice of this method depends on available information and processing power.

Method	# Refs	Dist	Angle	Complexity	Challenges
Trilateration	3	Yes	No	O(1)	Susceptible to Inaccurate Distances
Multilateration	$n \geq 3$	Yes	No	O(n^3)	Computational Complexity
Triangulation	3	No	Yes	O(1)	Requires Extra Hardware
Probabilistic	$n \geq 3$	Yes	No	$O(3d^2)$ (d = grid)	Computational and Space Complexity
Bounding Box	2	Yes	No	$O(n)$	Final Position Error

Table 3.3: Various Position Computation Algorithms (Boukerche, 2008, p. 316)

3.4.2.1 Trilateration and Multilateration

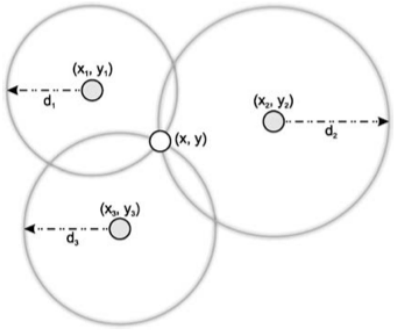


Figure 3.9: Trilateration Method - Figure from (Boukerche, 2008))

Trilateration is the process of locating a precise point in space by measuring distance to several points. By drawing several spheres with each radii being the distance to the point we're trying to locate, on a number of spheres from three (Trilateration) or more (Multilateration), it is possible to pinpoint the desired location, as shown in Fig. 3.9.
For example, if the point we're trying to locate has coordinates $P(x,y,z)$ and the other points used have coordinates $B(x_i,y_i,z_i)$. The equation for every one of these spheres will be:

$$(x-x_i)^2+(y-y_i)^2+(z-z_i)^2=r_i^2, i=(1,2,3,...,n) \tag{3.8}$$

Unfortunately, in real-world applications, the distance estimation inaccuracies as well as inaccuracies regarding the reference node's position make it so that the circles don't intersect into a single point, resulting in an infinite set of solutions.
When a large number of reference points are available, we can use the Multilateration approach to

compute the position estimation, resulting in an overdetermined system of equations - one where we have more equations than unknowns.

3.4.2.2 Triangulation

In triangulation, instead of the distances, we use information regarding the angles. The position is then computed using the trigonometry laws of sines and cosines.(Boukerche, 2008)
In the context of this thesis Triangulation will not be considered since we are using RSSI as our location parameter, and RSSI can be directly translated to a distance, making Trilateration and Multilateration algorithms a better fit.

3.4.2.3 Bounding Boxes

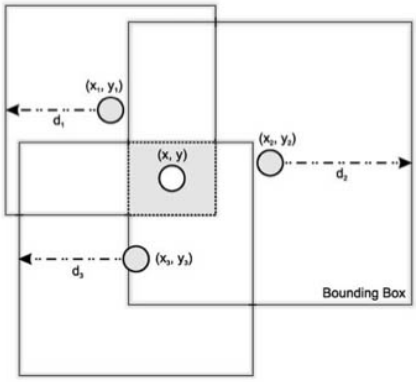


Figure 3.10: Bounding Box Method - Figure from (Boukerche, 2008))

Bounding Box is an alternative way of calculating the position based on the distance, proposed in (Boukerche, 2008, p. 321) using squares (instead of circles) to bound the possible location. For each reference node *i*, a bounding box is drawn as a square with the node as its centre with sides double the estimated distance, as seen in Figure 3.10.
The intersection of all bounding boxes can then be easily computed without the need for floating point operations by taking the maximum of the low coordinates and the minimum of the high coordinates of all the bounding boxes.
This method presents a larger error than that of it's competing methods but it requires a much smaller amount of processor resources.

3.4.3 Localization Algorithms

The localization algorithm is the main component of a localization system. It determines how the position and distance information will be manipulated in order to compute a valid position estimation for the target node. They can be classified into some categories (Boukerche, 2008, p. 334):

Distributed or Centralized Position Computation:
The positions of the nodes can be computed in a distributed way by the network nodes (self-positioning) or by a single central node (e.g.: a more powerful node);

With or Without an Infrastructure:
If there is no need for infrastructure, or if there is the need to redesign the previous infrastructure in order to allow the functioning of the localization algorithm;

Relative or Absolute Positioning:
The computed positions can be related to global coordinates (latitude, longitude) or related to a node or point of the network;

Indoor or Outdoor Scenarios:
If the system is used in indoors or outdoors scenarios.

One Hop or Multi Hop:
If all unknown nodes have direct communication with the beacon nodes (One Hop) or if a multi hop communication is needed.

In order to evaluate the performance and therefore the quality of the localization system, the following aspects can be used:

Mean Error and Consistence:
Defines the mean error of the position estimates and shows if the mean error is repeated in similar, but not equal, scenarios (consistence of the mean). This limits the usage of the localization system to applications where this level of inaccuracies is acceptable;

Communication Cost:
This identifies the algorithm complexity in terms of packets exchange;

Number of Settled Nodes: Establishes the percentage of network nodes that are able to compute their position. Ideally, all the nodes should be able to compute their position but this is not always possible;

Number of Beacon Nodes:
Determinates the number of beacon nodes required to make the algorithm work. A beacon node is a node that has information on its position, either by manual placement or by external means, such as GPS. Using beacon nodes is generally more expensive, so their usage should be minimized.

Performance can be affected by some network characteristics. It is important to bear in mind these traits as they will have different impacts in different situations. Some of these characteristics are:

Network Density:
In networks presenting a high density of nodes, we have smaller distances among the nodes resulting in lower errors in the distance estimation and thus a higher accuracy of the localization system.

Network Scale:

Increasing the number of nodes while keeping the network density increases the coverage area which in turn could result in a higher number of hops. Usually, a higher number of hops produces more inaccurate positions as a result.

Number of Beacon Nodes:

When beacon nodes are added to the system, the mean error of the localization system tends to decrease.

GPS Accuracy:

Although it is a fairly standard means of acquiring position coordinates, GPS does not provide perfect localization. Because most beacon nodes use GPS to attain their position, the GPS accuracy will severely impact the final position error.

3.5 GPS Coordinate Systems

Global Positioning System *(GPS)* is a space-based satellite navigation system providing location and time information, developed in 1973 by the U.S. Department of Defense. It is freely accessible to anyone with a *GPS* receiver and it can successfully calculate the position as long as it has line of sight with 4 or more satellites using the principles of *Trilateration* as described previously in this chapter.

The location is given by latitude, longitude and in some situations altitude, which is based on the height above the *WGS 84* geoid.

The World Geodetic System 1984 *(WGS 84)* is a Conventional Terrestrial Reference System, based on a consistent set of constants and model parameters that describe the Earth's size, shape, and gravity and geomagnetic fields. It is a right-handed, Earth-fixed orthogonal coordinate system as depicted in Figure 3.11. (Mularie, 2000)

WGS 84 is currently the standard U.S. Department of Defense definition of a global reference system for geospatial information and it is the reference system for the Global Positioning System *(GPS)*.

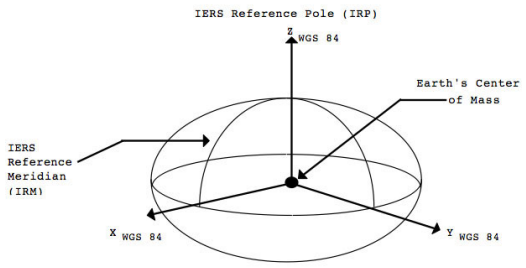


Figure 3.11: WGS 84 Coordinate System Definition (Figure from (Mularie, 2000))

Earth-Centered Earth-Fixed (ECEF) is a Cartesian coordinate system that represents positions as a (X,Y,Z) coordinate and has point $(0,0,0)$ defined as the centre of mass of the Earth.(Clynch, 2006)

3.6 Summary

In this chapter we expanded on the know-how required complete this concepts behind the completion of this dissertation. With this knowledge on hand, we can now start approaching related work in order to better understand how these problems are tackled.

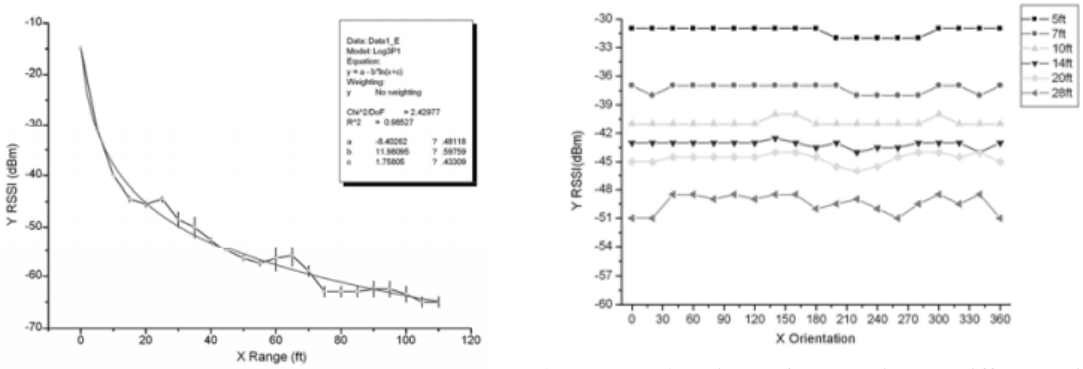
Chapter 4

State of the Art

After understanding our systems' requisites and the underlying background concepts, one must study the available technologies, striving to discover the most suitable means possible in pursuit of this thesis' goal.
In this chapter, we zero in on the previous work done in this field, specifying the current state of the art regarding UAVs and Wi-Fi Communications.We then focus our attention in localization system, specially the ones using Wi-Fi RSSI as Distance Estimation and the ones using UAVs.

4.1 Signal Propagation Models

In their study, Parameswaran et al. inquires as to the usability of RSSI as a Distance Estimation parameter, conducting practical experiments as to the variability of the signal and its conversion to a distance metric. Their results were less than encouraging for our work, as they found RSSI to be a less than optimal performer in localization algorithms.
Wu et al. presented a study, in 2008, regarding the characteristics of the RSSI signal reaching some very interesting conclusions regarding the behavior of the signal in the time and frequency domain: *"(..)(1) There is no relationship between the changing of RSSI signals and sampling time. (2) When collecting data in a wide-open area (field, rooftops), changes in RSSI signals are still evident, but can decrease drastically under the affects of objects in its path. (3) In time domain and frequency domain, RSSI signals do not have a periodic phenomenon. (4) RSSI signals' variance and its strength are not directly related to each other, but they are individually depended on the environment complexity.(...)"*
Fang et al. studied the variability of the RSSI signal in a wide manner of ways, including studying the signal variability outdoors with a non obstructed line of sight between receivers.



(a) RSSI versus Distance

(b) RSSI Values in a 360º Rotation at Different Distances

Figure 4.1: RSSI Variability Tests (Extracted from (Fang et al., 2010))

4.2 Wi-Fi Based Localization Systems

There have been several studies approaching Wi-Fi Based Localization, using various methods such as statistical or probabilistic approaches, by training the system while offline, building a database based on signal-to-location and the online sensing to find the Wi-Fi Node.

Another procedure entails using smart antennas that leverage the signal's direction in order to triangulate the Wi-Fi transmitter position. The problem with this technique is that, although it does not require offline training, it does pose the problem of signal reflection and multi-path fading. Other solutions are focused mainly on indoor locations, due to the problems cited above. These include using radio-maps obtained a priori, fingerprinting the RSSI previously, using hierarchical *Bayesian Sensor Models*, associating complimentary sensors such as odometers, among others.(Ibrahim and Ibrahim, 2010)

There are several differences between trying to achieve Wi-Fi location Indoors and Outdoors. For the purpose of this thesis, **we will focus exclusively on outdoors location as this dissertation focuses singularly on non-urban, outdoor** areas where we won't have the problem of obstacles and loss of line-of-sight between our access points.

In 2011, Zhang et al., searching for a simple, *Common-Off-The-Shelf* method to locate Wi-Fi APs, posed that it would be possible to do so using smartphones by placing it adjacent to the body of the user, and rotating himself, thus emulating a large directional antenna. This proves to be very effective on determining the direction of the APs, allowing the user to take an iterative approach pinpointing the Wi-Fi mode. Li et al. discusses Trilateration as a possible method of pinpointing APs by following two separate steps: using a signal propagation model to convert RSSI to distance between nodes, and after that using least-squares, or other methods such as a geometric method to compute the location.This strategy seems to adapt best to our necessities, because of its simplicity and yet, high effectiveness in our mission environments. The two steps will be discussed in greater detail in the following sections. The study (Chung et al., 2007) attempts to track a user position in both indoor and outdoor environments, using RSSI as a Distance Estimation, dividing the distance

estimation algorithm into a Deterministic Phase and a Probabilistic Phase in order to achieve better accuracy in their estimations.
A more recent study, (De Cauwer et al., 2010), shows another attempt at attempting a localization based on the RSSI value in a Wireless Sensor Network. They use several methods for their computations and register the mean error measured in practical scenarios for each of them. This is a particularly interesting study as it provides performance information regarding Position Computation Algorithms. In the same subject a study (Wang et al., 2009) was made in the previous year using a multilateration algorithm to compute a location system for Wireless Sensor Networks. An identical approach is used, but the conversion of the RSSI values to a distance metric is previously calibrated for the area of in which the sensor network is deployed.

4.3 UAV Based Localization Systems

There are studies regarding UAV Based Localization Systems, and we will further study and analyze them in order to have a better sense of the common approaches to this issue.
In (Frew et al., 2005), we are presented with a networked communication, command and control architecture that has a stage focused on localizing radio sources using a cooperative UAV team. Although it is a very interesting approach, it uses multiple UAVs to compute the position of the radio source, whereas our approach resorts to a single UAV.
In 2010, (Wagle and Frew, 2010) presented a particle filter approach to solving radio sources localization using only RSSI measurements, using UAVs to test their algorithms. They achieved localization with success but at the cost of a large localization error. They attribute some of this error to the UAV motion, something we will also have to take in account during our project.
In (Mao et al., 2007), we can see an attempt of using an Extended Kalman Filter to provide locations for the UAVs in cooperative flights using inter-UAV measurements. A location error of $40m$ is achieved. While this approach is quite interesting, we cannot extract much from it except that perhaps a similar localization algorithm using an Extended Kalman Filter is worth investigating.
We see the use of Extended Kalman Filters again in (Rullan-Lara et al., 2011). However, this study proposes the uses of time difference of arrivals (TDoA) to calculate the distance estimations. This method is not applicable in our system as it would require the installation of additional hardware on our UAV.

4.4 Summary

There are several studies related to RSSI variability, diverging in opinions regarding its usability, and several uses of this measure as a distance estimation in localization systems. Several of these even use UAVs as the mobile beacon in the localization process. There is however no studies regarding this particular challenge and as such we believe it is quite an interesting subject to approach, now that we have our know-how well based on related works done in the field.

Chapter 5

Implementation

This chapter will describe the steps taken in the development of our system in order to fulfill the requirements explained in Chapter 2. We develop a plug-in to the Neptus framework - programmed and implemented in Java - describing its implementation sparing no details. In further sections we describe each system, conferring to all the requirements we've planned during the first chapters of this dissertation.

5.1 Overview

We've described the approach to solving with the help of a high level functional architecture.
In pursuance of our goals, we've created a Neptus Plugin with the following capabilities:

- Antenna Localisation - Capable of locating one or more antennas during UAV flight;
- Antenna Range Estimation - Capable of estimating the range of antennas in conjunction with their position information;
- Save & Load - Capable of creating logs that can be later used, in order to predict range and/or link quality offline;
- Graphical User Interface - Graphically presents the results to the operator;

5.2 Functional Architecture

In Figure 5.1 we have the functional architecture for our complete Neptus Plugin. The white blocks are the main blocks and the foundation of our system while the blocks represented in blue are a sub-blocks of our localization algorithms system and are accessory functions to increase our system's performance. These are considered part of the main system blocks as well. The blocks represented in gray are external agents to our system, and the blocks represented in green are functions that are exclusive to Neptus and its functionalities. We will now make a description of set of

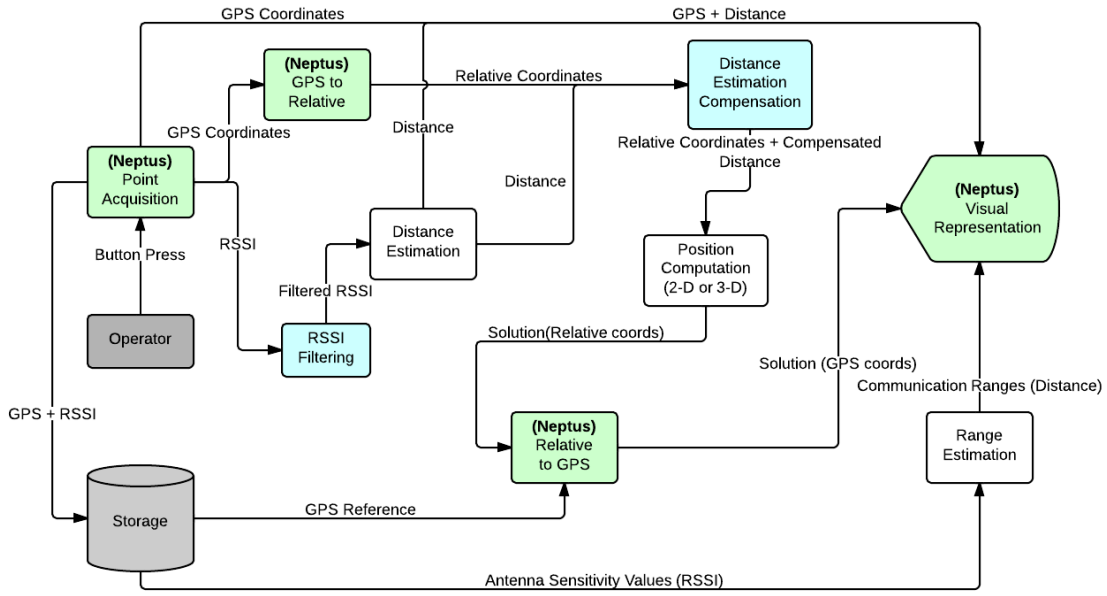


Figure 5.1: Functional Architecture

blocks, and enumerating their inputs and outputs and briefly outlining their functions:

Main

Distance Estimation: This block has as input a RSSI value. Its function is to convert values from RSSI to Meters;

Position Computation: This is where the position calculation takes place, having relative coordinates and a distance inserted.

Range Estimation: Has a functionality similar to the Distance Estimation block except it has for inputs a set of RSSI values, instead of just a single value, returning a set of distances. These distances are the communication ranges for a given antenna;

Localization Algorithm : This is comprised by two sub-blocks, RSSI Filtering and Distance Estimation Compensation. The first one has for input a series of RSSI values and for output a filtered RSSI value. On the latter one, we input a set of distances and relative coordinates and output the same relative coordinates with adjusted distance. This function will compensate underestimations and overestimations by our Distance Estimation block;

Neptus

Point Acquisition: Here, on the operator's command, we register the current UAV position and it's RSSI value related to the antenna (or antennas) and output the GPS coordinates and the RSSI;

WGS-84 to Relative: On this block, the conversion between GPS values and Relative values takes place having for inputs two sets of GPS coordinates: the GPS coordinates of a reference point, and the GPS coordinates of the point we want to convert. This function is particularly useful in order to convert the GPS position of the reference nodes to a position based on a cartesian referential in order to input said cartesian positions into the Localization System;

Relative to WGS-84: Here we have the inverse operation, converting relative coordinates back to GPS coordinates using as inputs the point we want to convert in relative coordinates and the same GPS Reference point as before. This function is used to convert the positions given by the Localization System back to a GPS position in order to be able to draw them in the Neptus georeferentiated map;

Visual Representation: This block takes care of the visual representation on the Neptus geoferentiated map. For that reason, it can have as many inputs as desired as long as they are GPS coordinates and distances;

External

Storage: Represents the hard-drive of the computer;

Operator: Represents the Neptus operator.

5.3 Algorithms

This section is dedicated to covering the algorithms computed behind the Distance Estimation, Position Computation, Localization Algorithm and Range Estimation blocks, and the theoretical foundation supporting them.

5.3.1 Distance Estimation

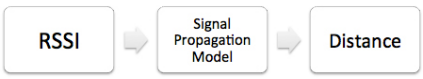


Figure 5.2: Signal Propagation Model Flowchart

As we can see in Figure 5.2, this algorithm processes one input, RSSI in *dBm* (power ratio in decibels of the measured power referenced to one milliwatt), resulting in one output, the distance

in meters.
We've discussed various ways to achieve this conversion in Sub-Section 3.3.2, and we've seen that the one that better suits our needs for this particular body of work is the *Free-Space Propagation Model* since, in most cases, the antennas are placed at ground height, making the *2-Ray Model* unnecessary for our calculations.
Calculating an estimation of distance from the RSSI values is just a matter of feeding said values to the Friis' Free Space Equation (Equation 3.3). Converting said equation to the measurement units we are using - meters for distance and kilohertz for frequency - and to produce a distance, we get:

$$RSSI(dB) = 20log_{10}(d) + 20log_{10}(f) - 27.55 - (\sum G - \sum L) \tag{5.1}$$

$$d(m) = 10^{(27.55 + RSSI - 20log_{10}(f) + (\sum G - \sum L))/20} \tag{5.2}$$

With d as the Distance in *meters*, *RSSI* as the Received Signal Strength in *dBm*, f as Frequency in *kilohertz*, G as the Gains of the receiver and transmitter antennas and L as the Signal Losses.

Observing the Equation 5.2, we notice that while the values of the receiver and transmitter antenna gain (G) are constant and provided by the manufacturer of the antennas, the value of the Signal Losses (L) are not.
Signal Losses are affected by several factors, such as:

- UAV Antenna Position;
- Aircraft Attitude;
- Physical Properties of Electromagnetic Waves (described in section 3.3.1);
- Terrain Topography;
- Many other unpredictable factors.

So, as we can see, there are several factors that come into play when calculating distance based on the *RSSI* value. If you add the fact that the *RSSI* as measured in the hardware is also highly unstable, the result is a very unpredictable system.
Some sort of calibration is therefore recommended, so we've decided to calculate the sum of gains and losses using a reference distance, d_{ref}. Using the same equations as before, we get:

$$(\sum G - \sum L) = RSSI(dB) - 20log_{10}(d_{ref}) - 20log_{10}(f) + 27.55 \tag{5.3}$$

As we've seen in Section 3.3.2.2, and in Equation 3.5 this model only applies for distances greater than the *Fraunhofer Distance* so an adequate d_{ref} must be chosen. The total value for the

system gains and losses is then inserted into Equation 5.2 and the distance is estimated.

5.3.2 Position Computation

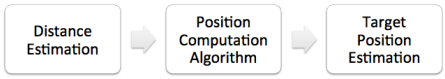


Figure 5.3: Position Computation Flowchart

Now that we have an estimation of the distance between the UAV and the antennas connected to it, we can compute their position using a position computation algorithm. Since we had various algorithms to choose from, we decided to implement three of them and then perform various performance tests to decide which one would be the most adequate. The methods chosen were: the *Least Squares* Method, the *Bounding Boxes* Method and the *Area of Intersection* Method and their implementation will be detailed in further sections.

5.3.2.1 Least Squares

This method is based on the error variance. It's goal is to find a value of the error variance that is between the estimated values and the experimental values. The simplest case of this estimator requires linear equations on the observation matrix. As such, this requires us to linearize the circle equations, seen in Equation 5.4. Following the same linearization method shown in (Murphy and Hereman, 1995), which uses the $j'th$ equation of Equation 5.4 as the linearization tool. By adding and subtracting x_j and y_j to all other equations, this leads to:

$$(x-x_i)^2+(y-y_i)^2+(z-z_i)^2=r_i^2, i=(1,2,3,...,n) \tag{5.4}$$

$$(x-x_j+x_j-x_i)^2+(y-y_j+y_j-y_i)^2+(z-z_j+z_j-z_i)^2=r_i^2, i=(1,2,...,j-1,j+1,...,n) \tag{5.5}$$

Simplifying, with $r_j=r_i$ which is the distance between the reference nodes and the target node and d_{ij}, which is the distance between the two reference nodes, we have:

$$(x-x_j)(x_i-xj)+(y-y_j)(y_i-yj)=\frac{1}{2}(r_j^2-r_i^2+d_{ij}^2) \tag{5.6}$$

It is irrelevant which equation is used as a linearization tool, so we'll arbitrarily choose $j=1$, which is to say, we choose the first reference node and compare it to all the other reference nodes

resulting in a linear system of equations with $n-1$ equations and two unknowns(Reichenbach et al., 2006).

$$\begin{aligned}
&(x-x_1)(x_2-x1)+(y-y_1)(y_2-y1)=\frac{1}{2}*(r_1^2-r_2^2+d_{21}^2)\\
&(x-x_1)(x_3-x1)+(y-y_1)(y_3-y1)=\frac{1}{2}*(r_1^2-r_3^2+d_{31}^2)\\
&\ldots\\
&(x-x_1)(x_n-x1)+(y-y_1)(y_n-y1)=\frac{1}{2}(r_1^2-r_n^2+d_{n1}^2)
\end{aligned} \tag{5.7}$$

We can now write the problem in the $Ax=b$ form, and proceed to solve it using the *Least Squares Method*.

$$A=\begin{bmatrix} x_2-x_1 & y_2-y_1\\ x_3-x_1 & y_3-y_1\\ \ldots & \ldots\\ x_n-x_1 & y_n-y_1 \end{bmatrix},x=\begin{bmatrix} x-x_1\\ y-y_1 \end{bmatrix},b=\begin{bmatrix} \frac{1}{2}*(r_1^2-r_2^2+d_{21}^2)\\ \frac{1}{2}*(r_1^2-r_3^2+d_{31}^2)\\ \ldots\\ \frac{1}{2}*(r_1^2-r_n^2+d_{n1}^2) \end{bmatrix} \tag{5.8}$$

Solving the *Least Squares* problem is a matter of applying the *Euclidean Norm*, which minimizes the sum of the squares:

$$\underset{x\ \varepsilon\ \mathfrak{R}^n}{Minimize}\ \|Ax+b\|^2 \tag{5.9}$$

In turn, this equals to solving the system described in Equation 5.10:

$$x=(A^TA)^{-1}A^Tb \tag{5.10}$$

After finding the final equation, it is just a matter of implementing it. Both 2-D and 3-D versions were implemented as it is just a matter of adding another unknown to the equation. The suitability of this algorithm will be evaluated in the next Chapter, with comparison to the other methods.

5.3.2.2 Bounding Boxes

Implementing the bounding boxes solution referred in Section 3.4.2.3 is a matter of, instead of drawing a circle with radius equal to the distance to the target node (d_i), drawing a square with side equal to twice the distance to the target node $(2d_i)$. The intersection is then computed without the need for floating point operations as it is as simple as finding the maximum of the low coordinates and the minimum of the high coordinates.
The centre point of that rectangle will then be given by:

$$(\widehat{x},\widehat{y})=\left(\frac{max(x_i-d_i)+min(x_i+di)}{2},\frac{max(y_i-d_i)+min(y_i+di)}{2}\right) \tag{5.11}$$

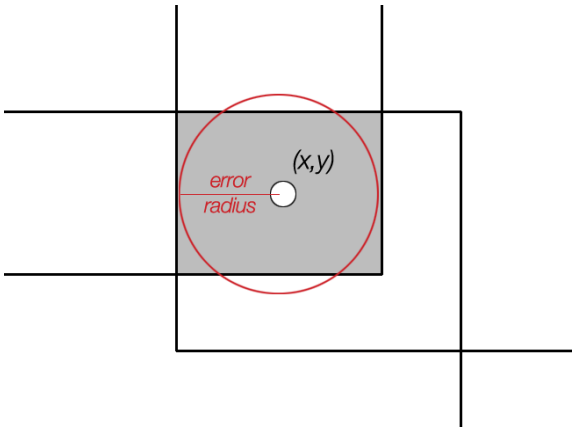


Figure 5.4: Bounding Box Error Approximation

As seen in Figure 5.4, the area of error would then be a circle of the same area as the rectangle of intersection, as a simplification, since every other method's error is displayed as the radius of a circle calculated as shown in the following equation:

$$A_{rectangle} = \pi * r_{error}^2 \Leftrightarrow r_{error} = \sqrt{\frac{A_{rectangle}}{\pi}} \tag{5.12}$$

As we can see, it is a very simple method and also very light in terms of processing necessities. This method was also implemented in 3-D, adding the Z coordinate without increasing the complexity, since it is just a matter of performing the exact same operations to the coordinate in the plane of zz.

$$(\widehat{x},\widehat{y},\widehat{z}) = \left(\frac{max(x_i - d_i) + min(x_i + di)}{2}, \frac{max(y_i - d_i) + min(y_i + di)}{2}, \frac{max(z_i - d_i) + min(z_i + di)}{2}\right) \tag{5.13}$$

5.3.2.3 Area of Intersection

This method is the classic Trilateration implementation, calculating the **Area of Intersection** between the 3 (or more) circles as we can see in the above figure. On a first approach, we will calculate the intersection between two circles, as in Figure 5.5

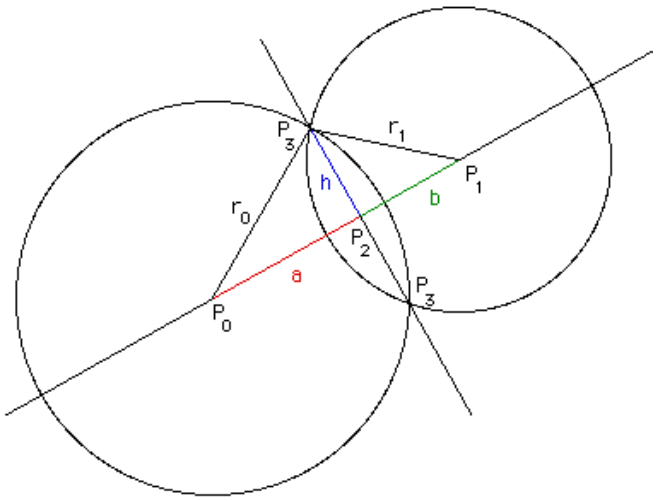


Figure 5.5: Calculating the Area of Intersection of Two Circles Figure from (Bourke)

The first step is calculating the distance between the two circles, given by:

$$d = \sqrt{(x_{P1} - x_{P2})^2 + (y_{P1} - y_{P2})^2} \tag{5.14}$$

- If $d > r_0 + r_1$, there are no solutions since the circles are separate;
- If $d < |r_0 - r_1|$, there are no solutions because one circle is contained in the other;
- If $d = 0$ and $r_0 = r_1$ then there are an infinite number of solutions, since the circles coincide.

Looking once more at Figure 5.5, considering the triangles $[P_0P_2P_3]$ and $[P_1P_2P_3]$, we can write:

$$a^2 + h^2 = r_0^2 \; and \; b^2 + h^2 = r_1^2 \tag{5.15}$$

Using $d = a + b$ we can solve for a:

$$a = \frac{r_0^2 - r_1^2 + d^2}{2d} \tag{5.16}$$

We can then calculate h using the following equation.

$$h^2 = r_0^2 - a^2 \tag{5.17}$$

So, the centre point of the area of intersection is given by:

$$\begin{aligned} x_{P2} &= x_{P0} + a(x_{P1} - x_{P0})/d \\ y_{P2} &= y_{P0} + a(y_{P1} - y_{P0})/d \end{aligned} \tag{5.18}$$

Now, by adding h to P_2, we can get the coordinates of P_3, the two intersection points.

$$\begin{aligned} x_{P3} &= x_{P2} + -h * (y_{P1} - y_{P0})/d \\ y_{P3} &= y_{P2} - +h * (x_{P1} - x_{P0})/d \end{aligned} \tag{5.19}$$

From the deductions above we can see that, when the circles intersect on one single point, i.e: $d = r_0 + r_1$, we get $a = r_0 = r_1$ and $h = 0$. With a value of zero for h, P_3 is a single point, and is equal to P_2. These deductions were extracted from (Bourke) and were the foundation for the following calculation of position.

After understanding how to calculate the intersection area of two circles, we can now tackle the challenge of calculating the intersection area between three (or more) circles.

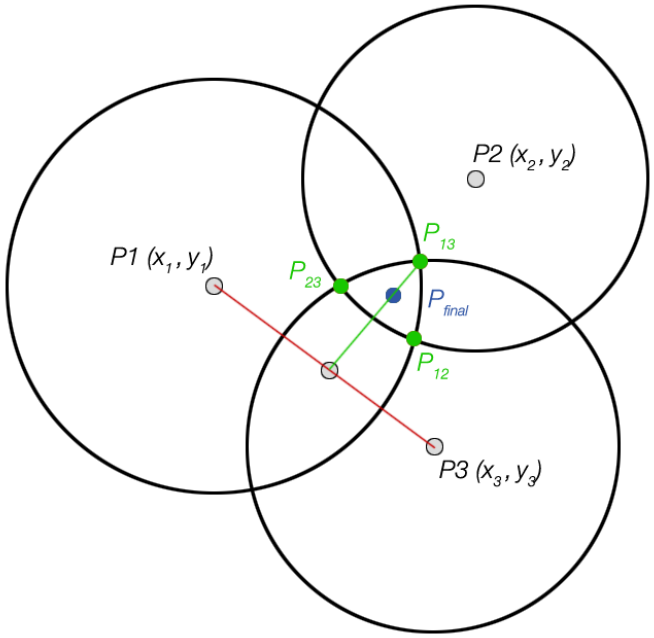


Figure 5.6: Calculating the Area of Intersection of Three Circles

Iteratively, we calculate the various intersection points for each circle pair. After calculating those points, the difficulty lies in selecting the points. In Figure 5.6 we can see the points we want to select, P_{12}, P_{13} and P_{23}. Thus we can see that the points we want will always be the ones closer in relation to each other.
The easiest way to achieve this is to sort them according to their distance relative to each other and to select as many unique points as the number of the circles we have. For example, in the case represented in Figure 5.6 we can easily see that the closest intersection points are in fact the ones defining the intersection area.
After having selected P_{12}, P_{13} and P_{23}, calculating P_{final}, the centre point of the area of intersection,

is trivially a matter of calculating the average of those three points as in the following equation:

$$P_{final}(x,y) = \left(\frac{x_{12}+x_{13}+x_{23}}{3}, \frac{y_{12}+y_{13}+y_{23}}{3} \right) \tag{5.20}$$

To estimate the error margin, the area of the intersection, much like in the *Bounding Boxes* method, we will approximate it to a circle with radius r_{error}. To create the best approximation, r_{error} is in fact the average of the distance between the centre point and every intersection point.

5.3.3 Localization Algorithm

The localization algorithm is the code segment that will decide how the previous algorithms are used in order achieve a valid position estimation for our target nodes, the antennas. In Section 3.4.3 we discussed a series of categories to classify *localization algorithms*. Following that reference, we can describe our algorithm in the following fashion:

Distributed or Centralized Position Computation: As we are using one single node that we can move around to simulate various nodes, we can classify it as centralized position computation.

With or Without an Infrastructure: There is no infrastructure involved or that needs to be redesigned when running our algorithm.

Relative or Absolute Positioning: The positions are computed globally, based on the *WGS-84* geoid, also used in GPS.

Indoor or Outdoor Scenarios: The system will always be used in outdoor scenarios.

One Hop or Multi Hop: All unknown nodes must have direct communication with the reference node, therefore it is a **one hop** system.

5.3.3.1 Distance Estimation Compensation

As we've established in Chapter 4, RSS-based localization systems often offer erroneous values, resulting in inaccuracies in the calculated distance. Some steps were then taken in order to increase the algorithm's performance.
On some cases, the distance estimation can underestimate the distance to a degree where our position estimation algorithm cannot compute a position, due to errors in the RSS measurements. As proposed in (De Cauwer et al., 2010) we developed a small algorithm that, if the position computation does not succeed, increases the estimated distance of each reference node by 10%. In situations like the one in Figure 5.7, it allows the computation of a position where it would not be possible.

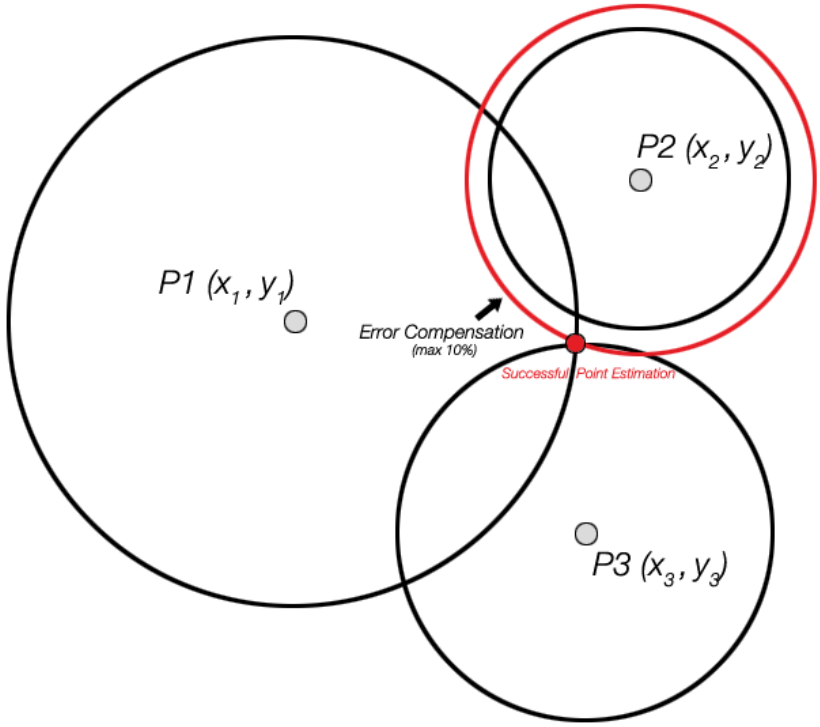


Figure 5.7: Estimation Error Compensation Scenario

5.3.3.2 RSSI Signal Filtering

As we've seen in the studies performed in previous chapters, RSSI signals always present a certain variance, depending on the antenna, environment conditions, etc. This, in turn, means that for the same position, two different measures can have different RSSI values.
In order to deal with this issue, we've decided to filter the RSSI values. We have decided to **implement a Moving Average Filter** - a finite impulse response filter - to provide us with a steady stream of RSSI values, while minimizing the impact of the outlier values.
We've decided to use **Brown's Simple Exponential Smoothing** (also known as Exponentially Weighted Moving Average) as a model because it is more responsive to changes on the recent past and better fitted for a rapidly updating system since, as we're trying to measure precise point values on a fast moving aircraft, the values quickly lose relevance as the plane continues its motion. For a series Y, it is calculated recursively as:

$$\begin{cases} S_1 = Y_1 \ , \ for\, t = 0 \\ \\ S_t = \alpha * Y_t + (1-\alpha) * S_{t-1} \ , \ for\, t > 0 \end{cases} \tag{5.21}$$

The α coefficient is a *"smoothing constant"* that can take values between 0 and 1.There is no

formally correct procedure for choosing α, so we've decided to choose $\alpha = 2/(k+1)$ and will further tune this procedure during testing.

5.3.4 Range Estimation

After pinpointing the antennas location with an acceptable degree of precision, the task befalls on estimating the range of that the UAV would have in the current environment. Using the exact same equation (Equation 5.2) that we calibrated in Section 5.3.1, but with predefined RSSI values we can calculate the distance. A quick examination of the antenna's data sheet tells us the sensitivity of the antenna for various transmission speeds based on the used protocol. This will allow us to map not only the maximum range for the antenna by using Equation 5.2 using the minimum sensitivity value, but also to be able to represent regions based on the network's throughput. This is very important to not only missions where high data transmission rates are critical, such as missions using real-time video or large sensor data packets, but also missions where it is important to have fast communication between ground station and aircraft.

5.4 Implementation

All of these algorithms need to have a practical application in the context of the missions endeavored by the *LSTS*. So, it was a logical step to implement it in their main supervision and control tool - Neptus - as a Plugin, a software component that adds a specific feature to an already existing software application.
This section details the implementation of the functional architecture blocks devised in Section 5.2 following the same order, with one section added in the beginning, dealing with the implementation of the inputs and outputs as Java classes.

5.4.1 Inputs and Outputs

In order to keep things streamlined and simple, when programming the input stage, we've combined the two inputs into a single input, implemented in a class called Shapes3D defined as: *Shapes3D(Point3D p, double radius)* where Point3D is defined as: *Point3D(double x, double y, double z)*. This was done so as to, when registering reference nodes, we could define a single *Shapes3D* Vector, with *Point3D p* being the coordinates of the reference point and *radius* being the RSSI value measured on that point.
The output stage will be done in a similar way except we've **created two different localization systems for two-dimensional and three-dimensional calculations**. In the two-dimensional system, the Antenna Position estimation is registered in the Solution class defined as: *Solution(Point2D p, double error)* with Point2D being *Point2D(double x, double y)*. Our 2-D Localization System will then return a *Solution* object, with *Point2D p* being the two-dimensional

coordinates of the antenna location estimate and *error* being the error calculated in the Localization System. Using the three-dimensional system, the 3-D Localization System returns an Antenna Position as an object of the *Solution3D(Point3D p, double error)* class, defined similarly as the above but using *Point3D* instead. As Neptus' GUI is a georeferentiated two-dimensional map, **we will only use the two-dimensional implementation** of the class *Solution*, but we kept the three-dimensional implementation for possible future work with UAV Localization Techniques. All the classes used as inputs and outputs are **serializable** in order to be able to record logs of the mission scenarios and solutions calculated for offline usage and storage.
When facing the task of detecting multiple antennas, the program runs every block for every antenna identifier - the MAC Address. Simply put, **if for one antenna, each system block is run once, for a n number of antennas, each block is run a n number of times.**
The implementation of the Localization System, Range Estimation and Distance Estimation will be addressed in further sections.

5.4.2 Distance Estimation

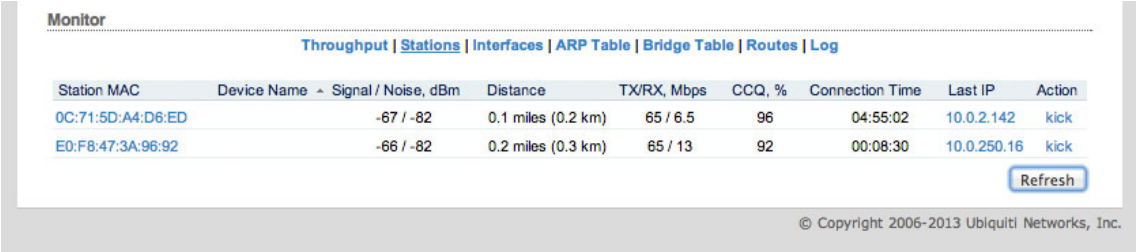


Figure 5.8: Router Configuration Webpage

There is no IMC message containing both the RSSI and a Source Identifier. That is to say, using the already defined messages, we have no way of knowing to what device the aircraft is currently connected. The antenna manufacturer, however, has made the RSSI values available as a table in the router configuration website, shown in Figure 5.8. This website, a common feature in most Wi-Fi radios, serves as a quick configuration method.
To overcome this hindrance we have decided to perform a *HTTP Query* to the router configuration website.
This was performed by writing a Java class that utilizes the *Apache Commons Libraries* to connect with the router configuration website website and search the connection table shown above for valid MAC Address and RSSI pairs, storing them for future use. This works by establishing a *HTTP Post Connection* to the router configuration page, and then parsing the *http//ROUTERIP/sta.cgi* table shown in Figure 5.8, storing all the values. Extracting the table values is simply a matter of parsing the *JSON* code into a Java-class.
This method allows us to, not only grab the *RSSI* value for one antenna, but also to grab the *MAC Address - RSSI* value pairs for everything connected to the aircraft's radio. This adds the capability

getting the distance to **various antennas simultaneously** and posteriorly compute their position. This routine was implemented on a separate class *HTTPSearcher* as a function *Map<String, Double> GetData(int grabs)*. The input value *grabs* is the buffer size to filter the RSSI values, discussed further ahead in Section 5.3.3.2. The function returns a *HashMap* containing the **antenna's MAC Address as key** and its associated RSSI as value.
After extracting the *RSSI* values, they were sorted according to their associated identifier - in this case, the *MAC Address* was chosen since it is already unique to each computer.
Equation 5.2 was implemented as a function *double calculateDistance(double levelInDb, double freqInMHz)*, with inputs as the RSSI value in dBm and the frequency in MHz, returning a *double dist* corresponding to the estimated distance in meters.

5.4.3 Position Computation

5.4.3.1 Least Squares

This method was implemented in the Plugin in both two-dimensional and three-dimensional variants.
The three-dimensional variant was implemented as a function *Solution3D TrilaterationCircles3D(Shapes3D[] circ)*, having for inputs an array of *Shapes3D(Point3D p, double radius)* and returning a *Solution3D(Point3D ap, double error)* object.
Similarly, the two-dimensional version was implemented as *Solution TrilaterationCircles(Shapes[] circ)*, having an array of *Shapes(Point2D p, double radius)* as input and a *Solution(Point2D ap, double error)* object as output.

5.4.3.2 Bounding Boxes

This method was also implemented in two-dimensional and three-dimensional alternatives.
The three-dimensional function is *Solution3D TrilaterationSquares3D(Shapes3D[] recs)*, also receiving an array of *Shapes3D* as input and returning a Solution3D.
The two-dimensional function, *Solution TrilaterationSquares(Shapes[] recs)* was implemented in the same fashion but with the two-dimensional variants of the inputs of the three-dimensional functions, *Shapes* and *Solution*.

5.4.3.3 Circle Intersection

This method was only implemented in two dimensions due to the added complexity of calculating the volume created by the intersection of three spheres. Therefore, only a single function was created, by the name of *Solution TrilaterationCirclesIA(Shapes[] circ)*, receiving an array of *Shapes(Point2D p, double radius)* and returning *Solution(Point2D ap, double erro)*.

5.4.4 Localization Algorithm

As detailed in Section 5.2, the localization algorithm used is comprised by two different system blocks, each with a function to complete in order to enhance the localization system's performance.

5.4.4.1 Distance Estimation Compensation

This mechanism was implemented in two functions, one for the three-dimensional position computation algorithms and one for the two-dimensional algorithms declared as *Shapes3D[] DistanceCompensation3D(Shapes3D[] shap)* and *Shapes[] DistanceCompensation(Shapes[] shap)* respectively. Both work in a similar fashion, by iteratively increasing the radius of each of the *shap* objects in the arrays and checking if there is a intersection between the spheres/circles until that condition is met or we reach an increment of 10% of all the original radiuses.
In situations where it still isn't possible to calculate an estimated position - that is to say, the 10% increased distance does not suffice - it is better to discard the current reference nodes, as we would be increasing the error of the localization algorithm by too large a factor.

5.4.4.2 RSSI Signal Filtering

This filter was implemented a simple function stated as *double EMACalculate(ArrayList<Double> values)* having for input an array of double values, the RSSI measurements and returning a single double value, the exponentially smoothed value of the RSSI signal. The number of values in the array is not to be left at random and will require tuning and further testing in the next chapter.

5.4.5 Range Estimation

The implementation of this section is tied in directly with the visual representation. We extracted the **sensitivity values for certain thresholds of transmission rates** and we used the *calculateDistance(double levelInDb, double freqInMhz)* function to compute distance values associated to those thresholds. We then programmed a simple function that **associates the various threshold distance values with different colors** and had them represented in the map as several concentric circles, all **centered in the antenna's position estimation** computed from the algorithms in the previous section.
The distances are calculated using the function *double calculateDistance(double levelInDb, double freqInMhz)* described previously, with the RSSI values used being the ones stated previously. This block is represented directly to the user and thusly is calculated directly in the *paint(Graphics2D g2, StateRenderer2D renderer)* defined by Neptus. We will approach this issue with greater detail further ahead.

5.4.6 Neptus

In this section we will detail the implementation of the Neptus-specific functional blocks, going into detail in regards to their usage.

5.4.6.1 Point Acquisition

When we press the button labeled *"Acquire 3D Point"*, several computations are made successively. Firstly, the function *HTTPSearcher.GetData(grabs)* is called with *grabs* being a determined buffer size. This will give us the MAC Addresses of the antennas connected to the aircraft and the RSSI values of the aircraft relative to each of them.
Subsequently, the current coordinates are registered in the object *Map<String, ArrayList<Shapes3D»* *point3DDataLogRAW*, which is a *HashMap* containing the **MAC Address identifier of the antenna as key** and a series of *Shapes3D* containing the global coordinates *(lat,lon,height)* and its associated RSSI value.
Current coordinates are stored in the object *GpsFix saveGpsFix* which is stored by **subscribing to the IMC Message *GPS Fix***. This message is a periodic message reporting the hardware reading of the GPS sensor. The message contains several packets of information, but the ones of interest for this use are:

Name	Abbrev.	Unit	Type	Description	Range
Latitude WGS-84	lat	rad	fp64_t	WGS-84 Latitude Coordinate	Same as field type
Longitude WGS-84	lon	rad	fp64_t	WGS-84 Longitude Coordinate	Same as field type
Height above WGS-84 ellipsoid	height	m	fp64_t	Height above the WGS-84 ellipsoid.	Same as field type

Table 5.1: GPS Fix Message - Coordinates (Table extracted from (LSTS, 2013)

Thus, the *point3DDataLogRaw* values are registered by adding new *Shapes3D* objects to the *Map<String,ArrayList<Shapes3D»* declared as a *Point3D* with the coordinates as $(X,Y,Z) = (saveGpsFix.getLat(), saveGpsFix.getLon(), saveGpsFix.getHeight())$ and *radius* as the RSSI Value.

5.4.6.2 GPS to Relative

As we discussed previously, our Localization System requires inputs in the form of **cartesian coordinates and a distance value**. As the IMC message pertaining GpsFix, seen on the previous section, only contains values in **global coordinates**, we must then perform a conversion before we can use the localization system. There is a set of functions designed by the LSTS to deal with these conversions, located in the *WGS84Utilities* package. The *WGS84Utilities.WGS84displacement* is a Java function developed by the *LSTS* found on the *WGS84Utilities* package. It calculates the

X, Y, Z distance between two points, given their *WGS 84* coordinates.

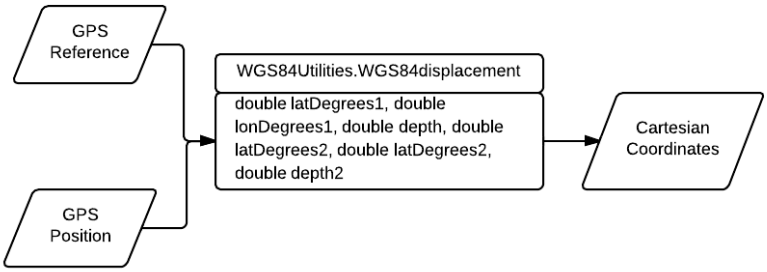


Figure 5.9: Calculating cartesian coordinates based on global coordinates

It becomes clear that we need some sort of baseline on which to establish the referential for our **cartesian coordinates**. This can be done by acquiring a GPS position on the launch point of the UAV and considering this first point our $(X, Y, Z) = (0, 0, 0)$ point.
This is done by storing the coordinates at the launch position, registered in the *saveGpsFix* into another object of the same class, *refGpsFix*. This object is then capable of being serialized in order to be stored for further use.
Using the *WGS84displacement* function with the coordinates from our *refGpsFix* object and *point3DDataLogRAW* objects will yield the **relative cartesian coordinates**, considering the *refGpsFix* as the point with coordinates $(X, Y, Z) = (0, 0, 0)$.
These values will then be stored onto an object of the same type as *point3DDataLogRaw*, *point3DData*.
We take advantage of being manipulating the data here to convert the RSSI values in *point3DDataLogRaw* to distance using the *calculateDistance(double levelInDb, double freqInMhz)* function and store the converted value into *point3DData*.
This way, **we separate the raw sensor data, registered in *point3DDataLogRAW* from the relative data, found in *point3DData***.

5.4.6.3 Relative to GPS

Converting the **cartesian coordinates back to global coordinates** is a matter of using a similar function located in the same package, *WGS84displace* that takes in as inputs a GPS reference, which we stored in *refGpsFix*, and a position in cartesian coordinates and returns the position in global coordinates.

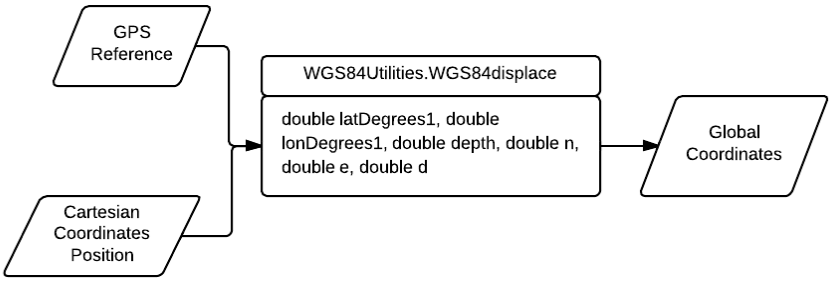


Figure 5.10: Calculating Global Coordinates based on Cartesian Coordinates

5.4.6.4 Visual Representation

The visual representation is done entirely using the *void paint(Graphics2D g2, StateRenderer2D renderer)* function. This function is called directly by the Neptus program, and by defining the *Graphics2D g2* object, we can then create objects that are painted on top of the Neptus georeferentiated map. To better understand, we present a small basic example here based on the implementation we followed when painting the acquisition points:

```
Graphics2D g = (Graphics2D) g2.create();
g.draw(new Ellipse2D.Double(centerPos.getX() - range, centerPos.getY() - range, range * 2,
                            range * 2));
```

This would draw on the Neptus georeferentiated map a circle with center with $Latitude = centerPos.getX()$, $Longitude = centerPos.getY()$ and radius *range*.

5.5 Usability

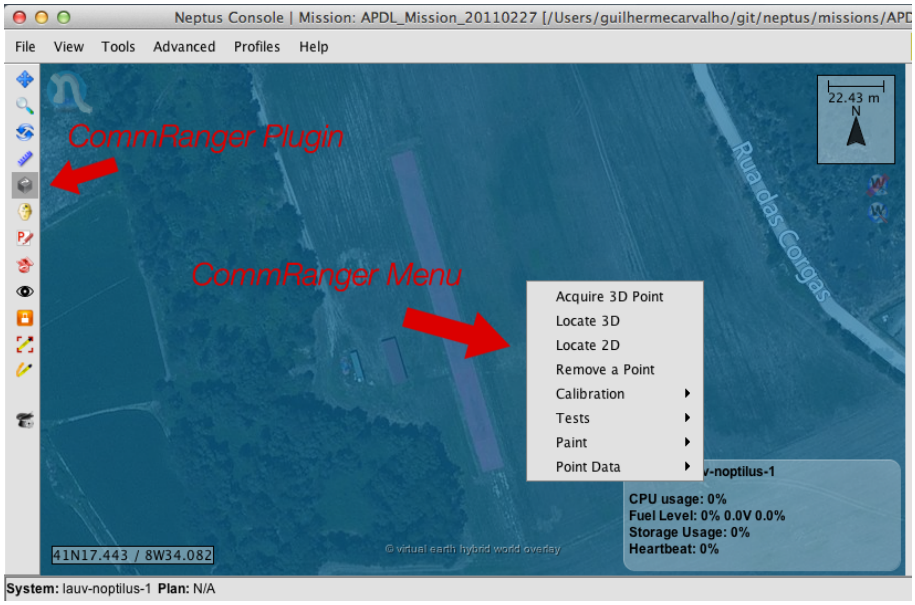


Figure 5.11: CommRanger Plugin Menu

The Plugin developed, dubbed *CommRanger*, has several functionalities which will be addressed in the following sections. With the Plugin selected on the toolbar, right-clicking anywhere on the map will present the CommRanger menu, shown in Figure 5.11.

5.5.1 Position Acquisition - Acquire 3D Point

Pressing the *"Acquire 3D Point"* button will register the current position of the UAV in GPS coordinates and perform the HTTP query to gather the RSSI value for that point. The two values will then be saved and stored.

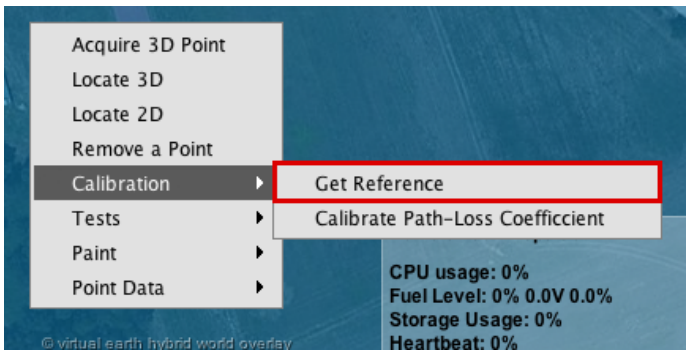
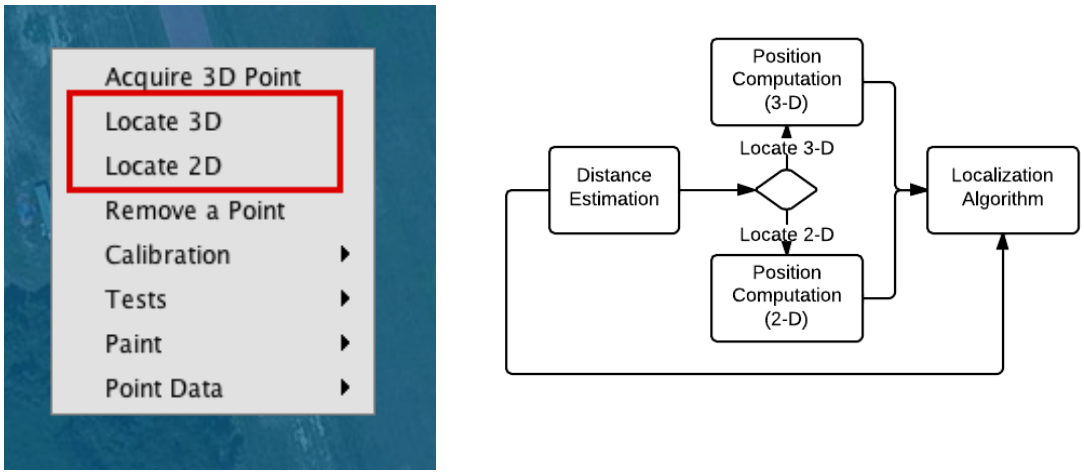


Figure 5.12: CommRanger Plugin - Get Reference

While GPS coordinates are necessary for visual representation, the Position Computation algorithms expect cartesian coordinates as an input. In order to overcome this hurdle, we **need to create a cartesian coordinate system in our mission area**. This will be done by defining the $(X,Y,Z) = (0,0,0)$ as our UAV launch point. You can define said point by pressing the *"Get Reference"* button, under the *"Calibration"* menu. This allows the following stored **acquisition points to be calculated with relative cartesian coordinates according to the launch point**. Conversions can then be made back and forth from global coordinates to relative coordinates using that reference point. An important part of any localization algorithm is segment how the reference nodes are chosen or discarded in order to compute the position estimation. Since our desired software of integration, Neptus, is a mixed initiative platform i.e., human operators in the control loop, we can successfully implement a **Manual Selection mode, where the operator decides which points to add and to remove**.
In Manual Mode, the task of selecting the reference nodes, i.e. the coordinate points that will be used to compute the antennas position, falls on the operator. The chosen reference points will be shown visually to the operator, and then the operator can judge the accuracy of the distance estimation and decide if that point is to be used in the position computation.
Ideally such mode would be deprecated in but as this is a work in development , this mode is very important in the testing phase of the program. In a final version, once the *Automatic Mode* is implemented, this mode would be redundant.

5.5.2 Antenna Location - Localization System



(a) Localization System Menu

(b) Implementation of the Localization System in the CommRanger Plugin

Figure 5.13: CommRanger Plugin - Localization System

Our localization system is of fairly straightforward implementation in the Neptus environment. As we can see in Figure 5.16 both three-dimensional and two-dimensional position computation

algorithms were implemented.
Implementing the three-dimensional was a direct application of the functions developed in Section 5.3.2, defining the function inputs as follows:

However, this is not the optimal way of locating the antenna and this method was implemented mostly due to the fact it was very straightforward to implement. Since on all of the situations tested, the **antenna was at ground-level**, we can implement a two-dimensional localization and improve the general accuracy of the algorithm. We do have a small hurdle to overcome, which is the fact that our distance estimation will always be three-dimensional. In order to deal with this drawback, **we must project the estimated distance into the** xy **plane**.

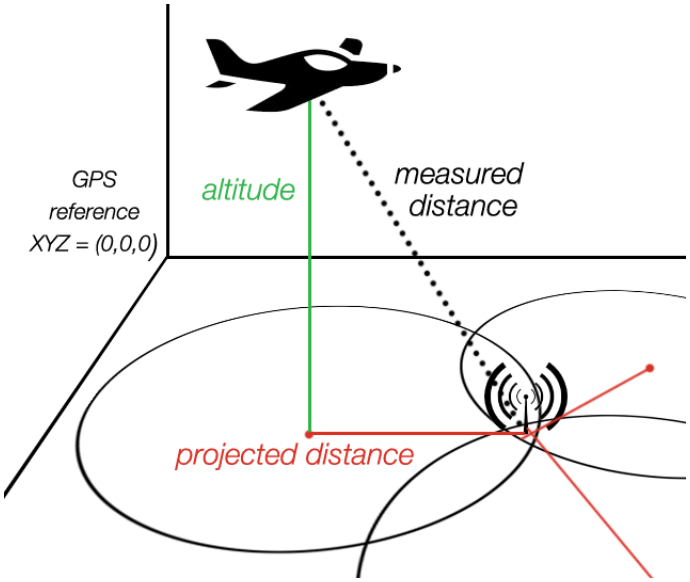


Figure 5.14: Calculating the projection on $z = 0$

If we use assume the antenna is placed at the same height we acquire our *GPS Reference*, projecting the estimated distance to the antenna's height is trivially a matter of solving:

$$dist_{projected} = \sqrt{dist_{real}^2 - altitude^2} \tag{5.22}$$

With $dist_{real}$ being the estimated distance, and altitude being the difference between the UAVs current altitude and the GPS Reference altitude, the inputs to the two-dimensional localization system can be defined as:

5.5.3 Range Estimation

The implementation of this section is tied in directly with the visual representation. We extracted the **sensitivity values for certain thresholds of transmission rates** and we used the function described in the distance estimation block (Section 5.3.1 to compute distance values associated to those thresholds. We then programmed a simple function that **associates the various threshold distance values with different colors** and had them represented in the map as several concentric

circles, all **centered in the antenna's position estimation**, computed from the algorithms in the previous section.

5.5.4 Paint - Visual Representation

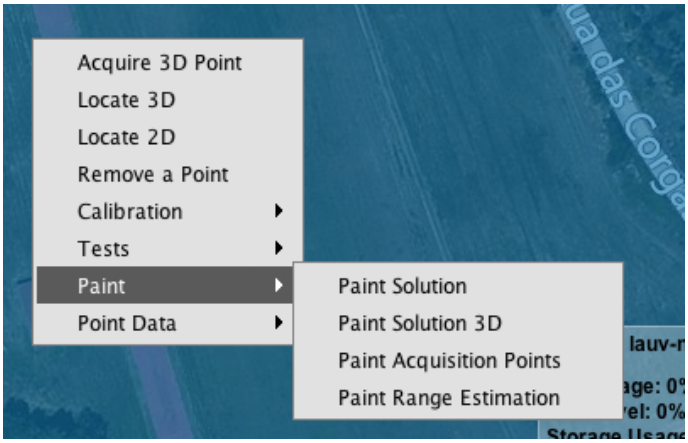


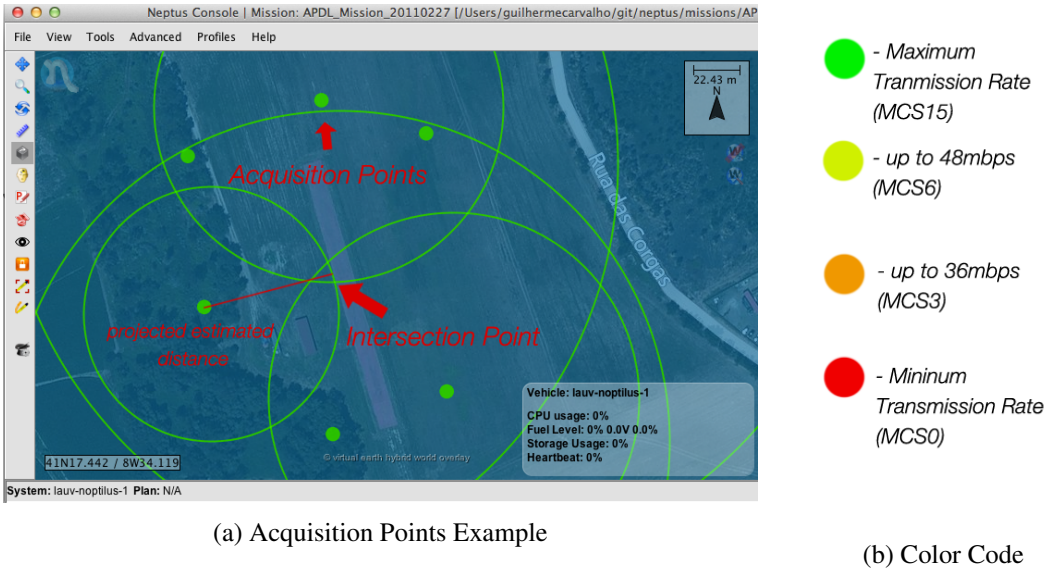
Figure 5.15: CommRanger: Paint Menu

The **Paint submenu**, pictured in Figure 5.15 turns the visual representations on and off, to act as a guidance for the operator. These features are of paramount importance because they enable the operator to have the **results of the computations displayed** quickly and directly on top of the Neptus mission environment, used to command and control the UAS.

5.5.4.1 Paint Acquisition Points

Visually representing the acquisition points and their estimated distances to the antennas had some challenges, since the visual representation used in Neptus is purely two-dimensional. We had no way of representing an altitude graphically to the operator. There is no need to represent the acquisition point's altitude for our purposes, as **we can just represent the latitude and longitude**, saving the altitude for the computations.
The distance estimation however is a different matter. We are representing the acquisition points so the operator can have a visual clue as to the distance estimation error in the measurements. If we were to represent the three-dimensional distance as the circle radius, in the two-dimensional environment, we would be considering the aircraft and the antenna to be at the same altitude and thus introducing an visual error to our representation. The Plugin might be able to compute a result at the aircraft's altitude but fail at the antenna's altitude, and we need our visual representation to exhibit that.
Our solution is therefore to project the distance estimation range much in the way we did in Section 5.5.2, using Equation 5.22.
Representing these projected distance estimations will allow us to see if we have an intersection

in the altitude where the antennas are placed.



(a) Acquisition Points Example

(b) Color Code

Figure 5.16: CommRanger: Paint Acquisition Points

Thanks to this visual representation, the operator also has a visual clue of how many points he has acquired, and their usefulness. In Figure 5.16a we can see that, some points have a very large distance estimation error. The operator can then choose to remove these points in order to reduce the output error on the localization algorithm.
In addition to representing the acquisition points and their estimated distance, i.e. the inputs to our localization system, we've also decided to **color code the acquisition points using the antenna's transmission rate sensitivity thresholds**. This is particularly useful as a confirmation of the Range Estimation calculated as we can **compare, visually, the estimated transmission rate threshold with the actual RSSI value obtained in that point**. The color code represents, for this antenna, the thresholds pictured in Figure 5.16b.

5.5.4.2 Paint Solution - Position Computation

To represent the various solutions, we've chosen different colors for each solution computed by each algorithm, as seen in Figure 5.17, representing the solutions computed during one of the field tests. It is also possible to see the actual GPS Location of the Antenna (represented as *"manta-3"*) as we've used an antenna with GPS receiver to determine the accuracy of our calculations.

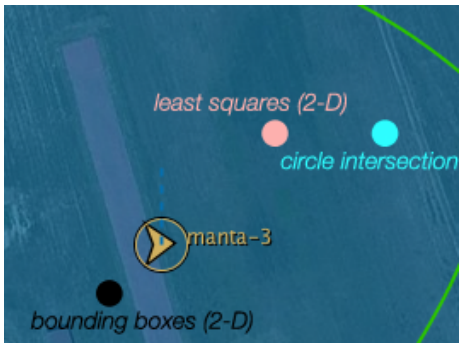


Figure 5.17: Solutions computed using the three different algorithms

This way allows the operator to **choose the solution most likely to be accurate and use it as the antenna position in the range estimation**.

5.5.4.3 Paint Range Estimation

After completing the localization process of the antennas (the *target nodes*) that is, having acquired a position with a mean error considered acceptable, we can now calculate display the estimated communication ranges in the Neptus environment.
Following the same color code we used for the *reference nodes*, pictured in Figure 5.16b we display the ranges calculated displayed as several circles with **radius equal to the maximum distance for a certain transmission rate**, centered in the antenna's localization estimation. In Figure 5.18 we have represented a range estimation computed during a flight test, where we can see the maximum range calculated as a *red circle* and the ranges for the several transmission rates as the other circles, with maximum transmission rate being the represented as the *green circle*.

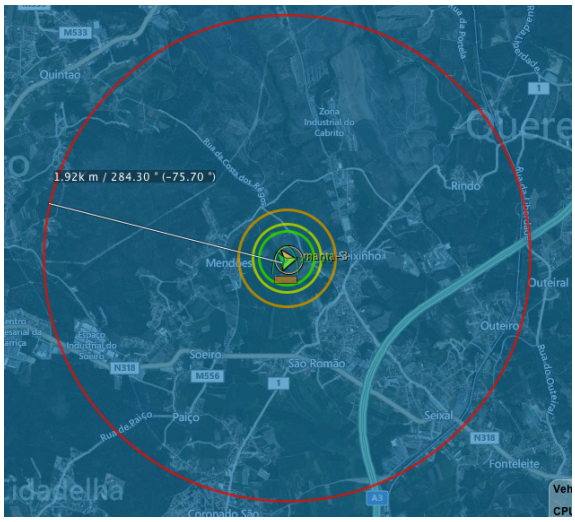


Figure 5.18: Calculated Range Estimation

5.5.5 Other Functionalities

There were several additional functionalities implemented in the *CommRanger Plugin* in order to enhance its usability by the operator such as the ability to *save and load* the points acquired, and a configuration file to save the various vehicle or antenna specific constants derived to be used in further occasions.

5.5.5.1 Test HTTP Conection and GPS Reference

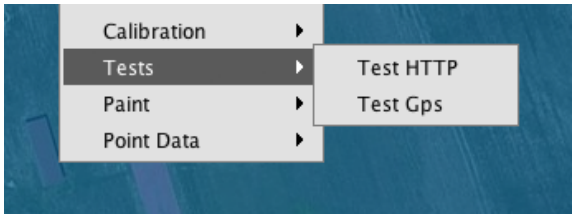


Figure 5.19: CommRanger: Test Menu

The test functionalities are present to enable the operator to check if the *HTTP Connection* is open and to see if the GPS Reference was correctly acquired. This allows the operator to correct eventual errors during setup in order to minimize the chance of connection failure or incorrect position acquisition during flight.

5.5.5.2 Calibrate Path-Loss

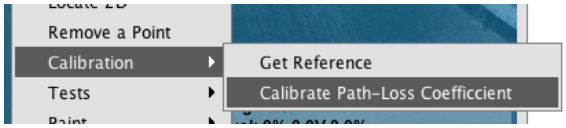


Figure 5.20: CommRanger: Calibration Menu

We implemented a routine that continuously measures RSSI in Section 5.3.1. The idea is to create a flight plan that performs circles around a known antenna location, keeping the distance to the antenna constant and then using said distance and the average RSSI during the flight in Equation 5.3 to calculate the system's associated gains-losses.
Implementing this function in the Plugin was not as straightforward as the Distance Estimation. We implemented a function as *calibrateGains(double dist)*. When this function is called with the input *dist* being the reference distance, d_{ref}, **the function begins registering RSSI values every second for 20 seconds**. After this cycle is done, the average RSSI value is calculated and used to input in Equation refeq:calibration.

5.5.5.3 Save and Load

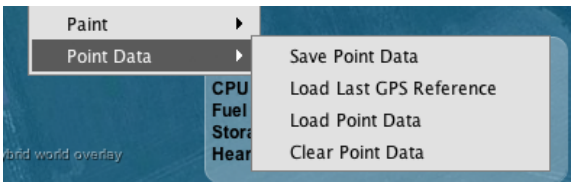


Figure 5.21: CommRanger: Save and Load Menu

The save and load functionality is of critical importance as it allows the operator to further test the algorithms in a development phase, and, in a final phase, to load range estimations when the circumstances are similar (e.g. the same location, same aircraft and the same antennas). The operator can choose to perform a series of options, detailed below:

Save Point Data: The save function *serializes* to a **.ser* file the stored values containing the GPS coordinates and associated RSSI value of the acquisition points and the GPS reference.

Load Point Data: The load function opens the serialized **.ser* file and loads the GPS reference and the acquired points and displays them in the map, allowing the user to compute the solutions once more using any algorithm desired.

Load Last GPS Reference : This function works similarly to the previous one except it only loads the GPS Reference. It is useful when the mission is still underway and you need to reset or restart the program, allowing you to keep getting points without having to re-acquire a reference point that, as we saw previously, must be done on the ground.

Clear Point Data: Clears all the data stored in the *point3DDataLogRAW, point3DData*, and *refGpsFix* objects. This enables us to reset the program and the stored values.

5.6 Summary

In this Chapter we've detailed everything done during the implementation phase of the project. We successfully created a solution to our problem, completely integrated in the workflow of the *LSTS*. In the next Chapter, we will submit the program created to several tests and simulations in order to evaluate its capabilities and plan the next steps in the development of this system.

Chapter 6

Tests and Simulations

This chapter will cover the simulations and field tests done during this dissertation. We will make considerations and validate the algorithms developed both in a theoretical sense and in a practical sense. We will also discuss the real world usability of this program and whether or not it fulfills its true purpose - increasing the awareness of the mission operator.

6.1 Simulations

We've devised a series of simulations in order to give us a better knowledge of the localization algorithms developed and how they will respond to different situations regarding the target nodes we are trying to locate and the amount of reference nodes used.

6.1.1 Simulator

We came up with a simulator that strives to recreate the errors present in our systems' inputs in a real scenario. This simulator was programmed in Java in order to use the exact same implementation of the algorithms that we plan to use on the final Plugin.

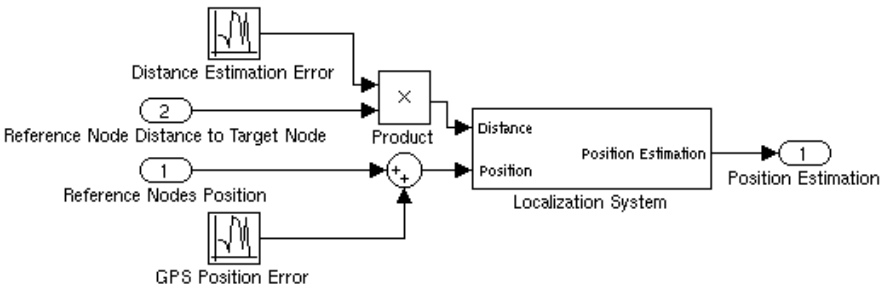


Figure 6.1: Simulator Block Diagram

The inputs of this simulator, **Reference Node Position** and **Reference Node Distance to Target Node** are created by defining a trajectory and selecting **Reference Nodes**. The two inputs will be the Reference Node's (X,Y) coordinates and its distance relative to the node we are attempting to locate: the Target Node.
We cannot expect the inputs to be free of error in practical situations, so in order to try and simulate the unpredictability of the RSSI we've introduced a random factor into the distance prediction. This is also to simulate the unpredictability associated with the motion of the UAV and other errors associated with the Distance Estimation.
We find errors not only on the RSSI values, but also on the position given by the GPS satellites. We've simulated this positioning error by adding a random value in meters to the X and Y coordinates of our reference nodes.
Since the inputs are associated with random factors, we compute a series of position estimations for each test, and calculate the mean error attained. The statistical data treatment was done in *Excel* by calculating the average, standard deviation and variance of the output error. Conclusions are then made based on said values.

6.1.2 Environment

The environment of our simulation is constituted by:

- A predefined trajectory from which to extract the reference nodes, simulating the path the UAV would make in the mission area;
- A target node, simulating the antenna;
- A flat, unobstructed area of $300x300$ meters.

We will simulate a certain chosen trajectory and acquire reference nodes along its path. The localization of the antenna is the focus of this simulation study and will vary according to our test plan.
As discussed previously, localization algorithms only produce correct position estimations if the **reference nodes are not collinear**, therefore, a circular trajectory seems like a good fit to our solution. Therefore we will consider a trajectory centered in $(X,Y) = (0,0)$ with a radius of 100 meters. A number of reference nodes contained in the trajectory can then be acquired in order to use as inputs to our localization system.
The localization system used in the simulations is exactly the one we implemented in the final Plugin version, and all the simulations will be done in the same conditions, using 3 to 7 reference nodes, with 5 computations for each set of nodes. 3 is the minimum number of nodes to compute a position, while 7 is a sufficiently large number to consider, but still small enough that it isn't a hindrance, computationally-wise.
The errors were introduced in an effort to simulate real conditions. The Distance Estimation Error is introduced to simulate faults with the signal propagation model and to simulate the inherent RSSI variance. We've done this simulation using generating random values from -10% to $+10\%$.

The GPS Positioning Error is introduced to represent the position error given by the GPS sensor. We've introduced values from -15 to $+15$ meters.

6.1.3 Test Plan

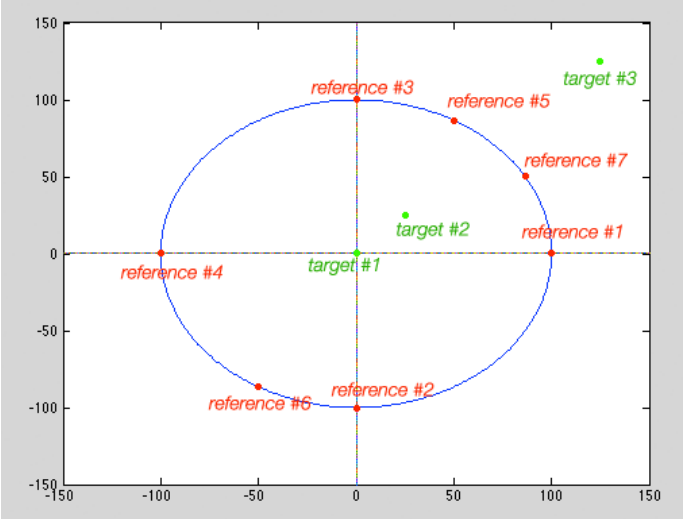


Figure 6.2: Reference Nodes and Target Nodes used for Simulation

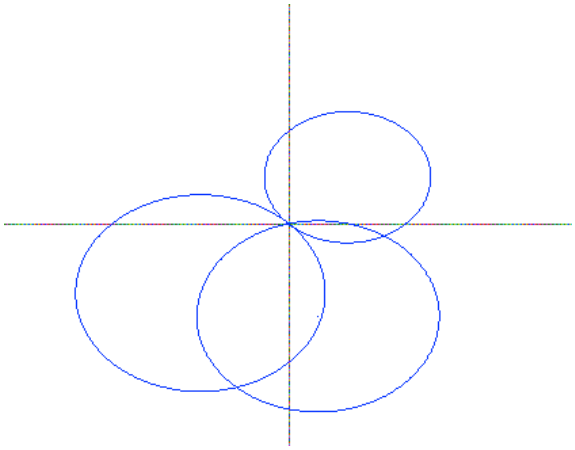
In the figure above we can see the flight routine (the blue circle) and the various reference nodes used for testing and the target nodes we tried to locate, presenting several levels of difficulty: a perfectly centered target node, a node somewhere in the center of movement and a node located to one side of every reference node.
We then devise a test plan, in pursuance of the goals dictated for this simulation.
First we will conduct a baseline test, which is to say, test the localization system without added errors.
Secondly, we will begin testing the system with added errors for each target node depicted in Fig. 6.2. This will be done as an iterative process, described as follows:

- Localization with Distance Estimation Error;
- Localization with GPS Position Error;
- Localization with both Distance Estimation and GPS Positioning Errors;

The number of reference nodes will increase by one at each cycle of the simulation process, starting at 3 reference nodes, and ending at 7 reference nodes with 5 computations made for each set of reference nodes.
The results will then be passed into Excel for statistical testing, as described previously and presented in graphical form for analysis.

6.1.4 Results

Since this project is centered around practical implementation, it is not a project that can benefit a great deal from simulations. The fact of the matter is that, in a perfect simulated environment, our Localization System will compute a location with little to no error, regardless of the number of reference nodes.
If the Distance Estimation is exact and has no associated errors, then the whole system is free from error as you can see in the following figure:



(a) Reference Nodes Simulation

Position Computation Algorithm	Position Calculated	Error Calculated
Bounding Box	$(0.207, -0.352)$	0
Circle Intersection	$(-4.618 * 10^{-16}, -7.105 * 10^{-17})$	0
Least Squares	$(-8.81 * 10^{-16}, -8.81 * 10^{-16})$	0

Table 6.1: Position Values Calculated

Figure 6.3: Baseline Test

As you can see in Figure 6.3 all three circles intersect in $X, Y = (0,0)$. All algorithms compute positions that are basically zero, except bounding boxes which, by the way it is calculated inserts a fair amount of error into the computations. Increasing the number of reference nodes will not present change the result when working with accurate Distance Estimations.

6.1.4.1 Target Node #1

This target node presents the best possible scenario, but it is a very unlikely situation for it would require prior knowledge of the target node's position, thus making the location attempt redundant. Nonetheless, it is a good baseline for us to being our simulations.
We start by simulating the localization algorithm introducing error in the distance estimation:

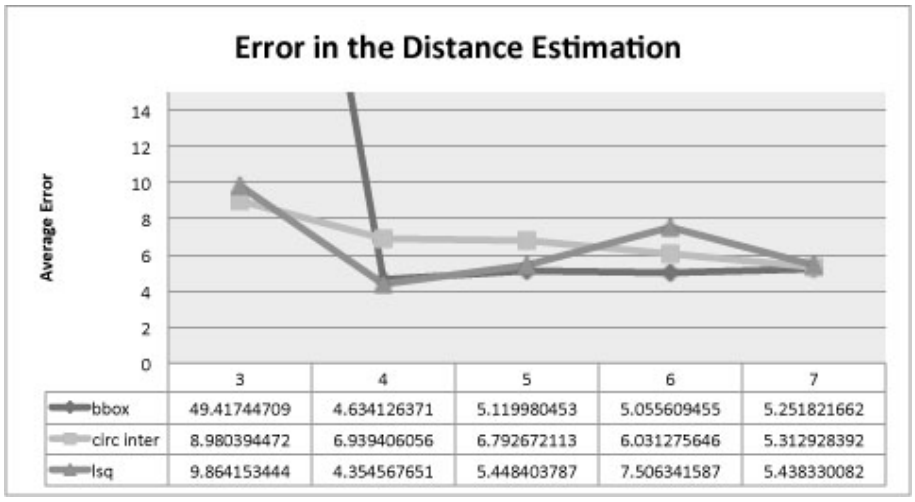


Figure 6.4: Average Error with Error in the Distance Estimation

As we can see, increasing the reference nodes does not necessarily yield an improvement. If the distance estimation error was always constant, we would see a steady decrease in error but, since we're simulating a system with several unpredictable factors **the error is also unpredictable**.

Next, we simulate the localization system with Error in the GPS Position to see how the system deals with errors in the Positioning of the reference nodes.

The Bounding Boxes Algorithm presented a very unusual error using only 3 reference nodes, not only during this test but propagating to every other test. We assume this to be, due to the way the algorithm is calculated described in Section 5.3.2.2, the reference nodes chosen as it computes perfectly acceptable solutions with 3 reference nodes in any other scenario.

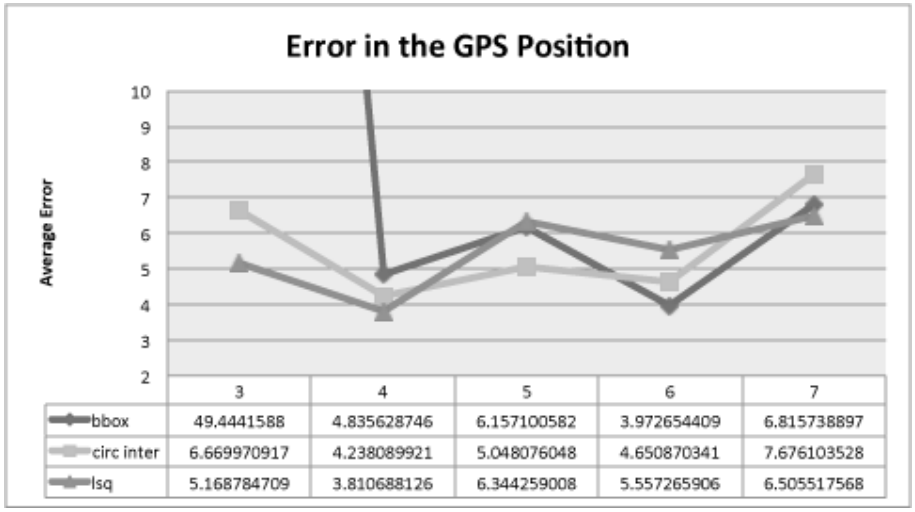
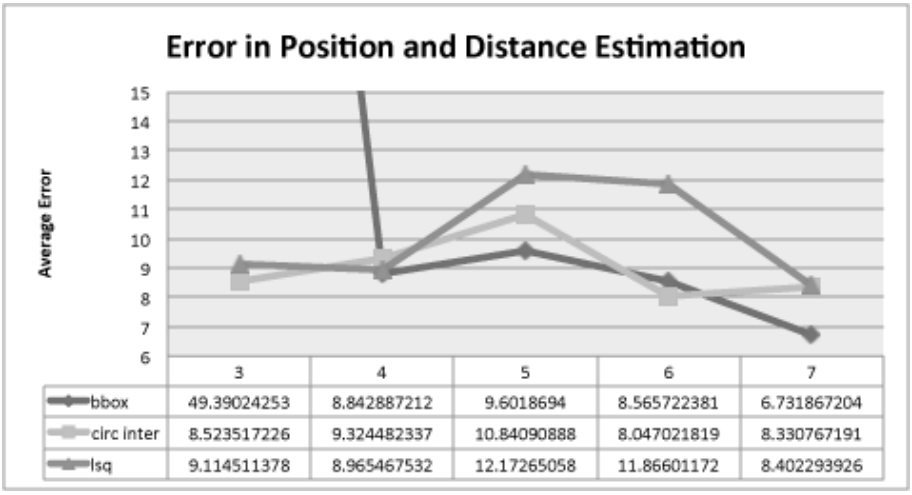


Figure 6.5: Average Error with Error in the GPS Positioning

Here we're presented with a higher variance depending on the reference nodes, but again the

resulting error does not change in any discernible fashion. Although here we can start to see that, the error reaches its minimum when using 4 reference nodes and 6 reference nodes. This suggests that the Localization System **performs better when the reference nodes are symetric** in relation to the target node.
When introducing error in both the Distance Estimation and the GPS Position, we start to approach the conditions we will face when using this program in real world situations.



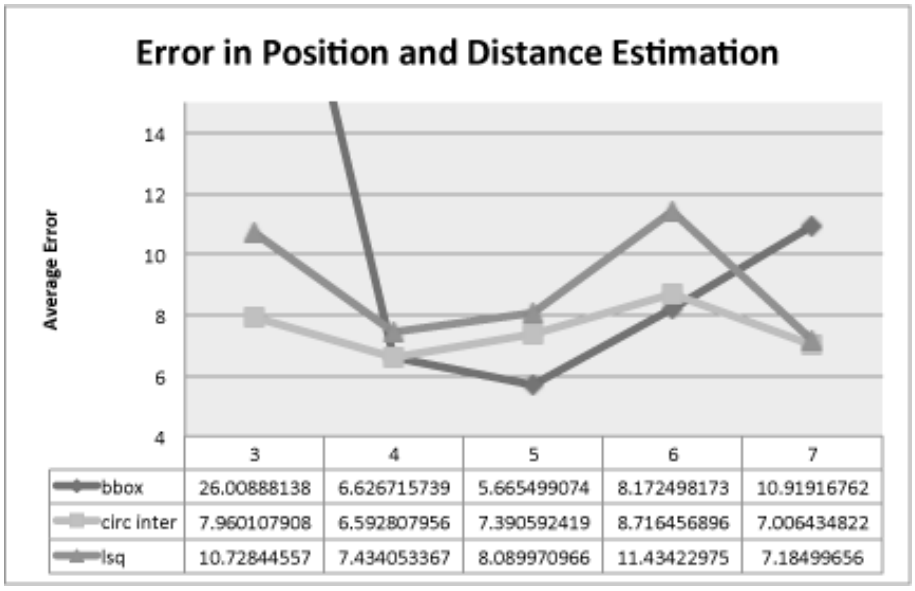
	3	4	5	6	7
bbox	49.39024253	8.842887212	9.6018694	8.565722381	6.731867204
circ inter	8.523517226	9.324482337	10.84090888	8.047021819	8.330767191
lsq	9.114511378	8.965467532	12.17265058	11.86601172	8.402293926

Figure 6.6: Average Error with Error in Position and Distance Estimation

Here, we face a much higher average error value for all algorithms. This was to be expected as the errors propagate through our algorithms and to the output of the localization system. Surprisingly, all algorithms had minimum error when using 7 reference nodes, thus indicating that **more reference nodes introduced to the system will result in a lower error**.
We can see here that one algorithm outperforms the others when in severe conditions: the Bounding Box Algorithm. It is the least sensitive to error and kept its average error approximately constant from 4 reference nodes on.
It is worth noting that when working with this amount of error, on some occasions the localization algorithm will not be able to compute a position. On every other test we had a 100% success rate but on this last one, the one with the biggest error margins, we managed an 83% **success rate**, with failures occurring using 6 reference nodes and 7 reference nodes. This is nothing out of the ordinary, and we took that into account when designing the program by **discarding failed attempts and acquiring new reference points**. These values are not to be taken for absolute as our sample size was not big enough, nor was it deemed necessary to have such a thorough approach to a system that will be inherently different in practical applications. So, we believe that in real testing we might come to different conclusions.

6.1.4.2 Target Node #2

Target Node #2 is located inside the circle described by the flight routine, but not centered. This is a much more plausible simulation of real circumstances. We conducted the same tests as for Target Node #1 and the same two tests performed similarly locating both target nodes. It is worth saying that, while the GPS Position Error had the same influence on both simulations, we found that the **error associated with Distance Estimation Error was marginally smaller** on all situations. This might be due to the distribution of reference nodes, with the area around Target Node #2 being more densely populated with reference nodes than the area around Target #1. This leads us to believe that **acquiring our reference nodes on the close vicinities of the target nodes might have a diminishing effect on the error of the location**.



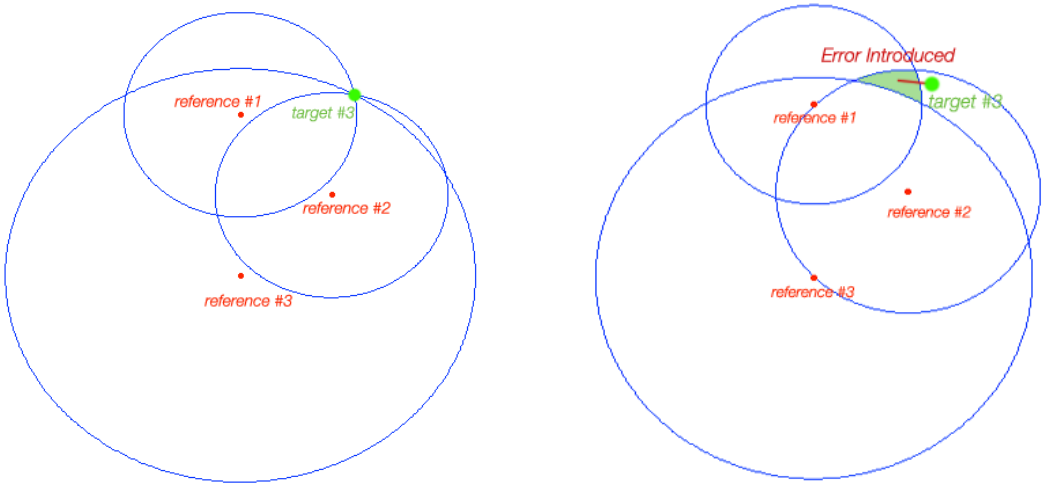
	3	4	5	6	7
bbox	26.00888138	6.626715739	5.665499074	8.172498173	10.91916762
circ inter	7.960107908	6.592807956	7.390592419	8.716456896	7.006434822
lsq	10.72844557	7.434053367	8.089970966	11.43422975	7.18499656

Figure 6.7: Average Error with Error in Position and Distance Estimation

We are presented with error around the same order of magnitude as with Target Node #1, which leads us to the conclusion that, for regardless of where inside the circle described by the flight routine the points are placed, the localization system behaves similarly.
It is worth noting that the Bounding Boxes algorithm associated error shown a crescent increase as we added more reference nodes, suggesting that perhaps for these circumstances, 4 or 5 reference nodes represent greater accuracy for our localization system.
While testing this system some of the attempts resulted in failed localizations, as with the previous simulation and, similarly, we registered an 86% **success rate** with failures present in the localization attempts with higher reference nodes. This suggests that, in real applications, we should **balance the number of reference nodes with the localization success rate** in order to keep the localization as efficient as possible.

6.1.4.3 Target Node #3

Locating this node will present a challenge to our localization system. All our algorithms strive to locate one point or the **area of intersection**. In situations like the one we have on our hands, every reference node is located to the same side of our target node. Ideally, this should have no effect on our localization system, but when Distance Estimation or Positioning errors come into play, the circles will intersect in suboptimal ways, as shown in the following figure:



(a) Target #3 Localization without introduced error (b) Target #3 Localization with introduced errors

Figure 6.8: Target Node #3 Localization

So, we expected that introducing errors in this kind of situation will be a much bigger challenge for our localization system.

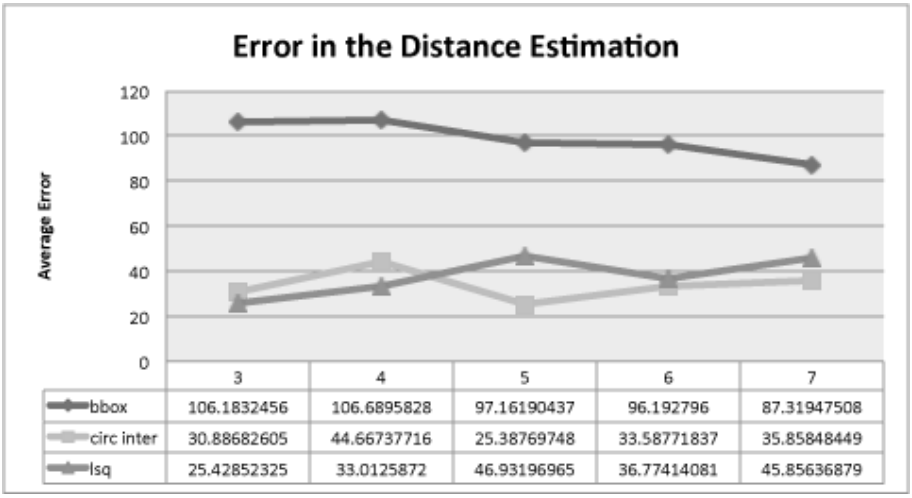
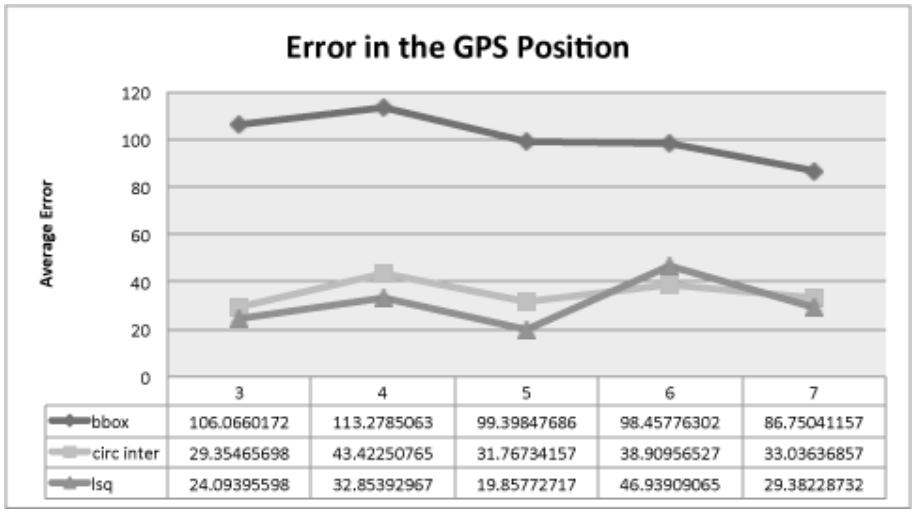


Figure 6.9: Average Error with Error in the Distance Estimation

Immediately we're faced with very different results than we found in the previous simulations. Every algorithm struggled to compute a precise location, and the accuracy results had an error of, in the best scenario, three times as larger as in any previous situation. To worsen the situation, in the Circle Intersection and Least Squares Algorithms, the standard deviation and variance were of such a magnitude that suggests that **the algorithms are very unstable in this scenario when present with error in the Distance Estimation**. The Bounding Box algorithm presented a very stable mean error value but it was immense when compared to the error demonstrated in other scenarios.



	3	4	5	6	7
bbox	106.0660172	113.2785063	99.39847686	98.45776302	86.75041157
circ inter	29.35465698	43.42250765	31.76734157	38.90956527	33.03636857
lsq	24.09395598	32.85392967	19.85772717	46.93909065	29.38228732

Figure 6.10: Average Error with Error in the GPS Positioning

In Figure 6.10 we encounter the same results. Substantially larger mean error values in this scenario. This time, however, the standard deviation and variance of the algorithms is smaller thus indicating that **the position computation algorithms are less unstable under Positioning errors**.

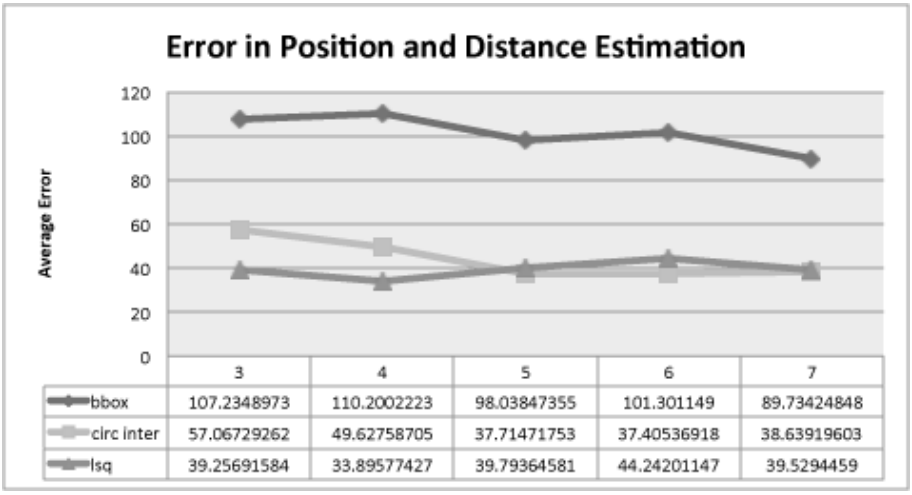


Figure 6.11: Average Error with Error in Position and Distance Estimation

Simulating this scenario with errors in both Distance Estimation and GPS Positioning yields results that are far from satisfactory but, as explained in the beginning of the section, were completely expected. In all simulations done in this scenario we found that **the position computation is unstable, therefore unreliable**. Not only that, but it also presents immense output error. The only algorithm that seemed to cope with this particularly challenging situation was the Bounding Box algorithm, at the expense of presenting an error ten times as large as in the situations simulated before. We consider this to be an extreme situation that should not happen often in the field as the operator should have a general idea where the antennas are placed and can create a maneuver that, at least, can achieve a situation similar to Target Node #2.

Regardless of these specific scenario simulations, the key tests to the localization algorithms will be the field tests, as this is a highly complex problem and difficult to fully comprehend through simulations alone.

6.2 Field Tests

In order to validate the solution devised in this dissertation, the program was tested in real UAV mission scenarios. Integrating with the existing LSTS toolchain, we strived to locate the antennas and predict a range estimation during operation, registering the data computed to analyze and extract conclusions. During these tests, antennas with GPS location were used in order to validate our location attempts and register the errors measured.
Unfortunately, as testing this project in real situations and scenarios is constrained by the possibilities of the LSTS schedule and the availability of the team, we could not submit the project to all the tests we deemed necessary. We could only conduct tests using a single antenna, but it is safe to assume that, as the measurements and computations are run in parallel for each antenna, it would work in a similar fashion with any number of antennas.

Not only was the testing constrained by the LSTS schedule, but also by the weather. Flight tests can only be conducted in meteorological conditions deemed safe for the aircraft, the operators and the equipment. Being that this project was developed during the autumn and winter, severe weather conditions were often present, further impeding the realization of more tests.
During the course of this dissertation developed, we had the opportunity to **test this project on two separate occasions**. The first was during RC Safety Training and Mapping tests, on the **first week of December 2013** and the second test was done during the **9th and 10th of January 2014**.

6.2.1 Test Location & Setup

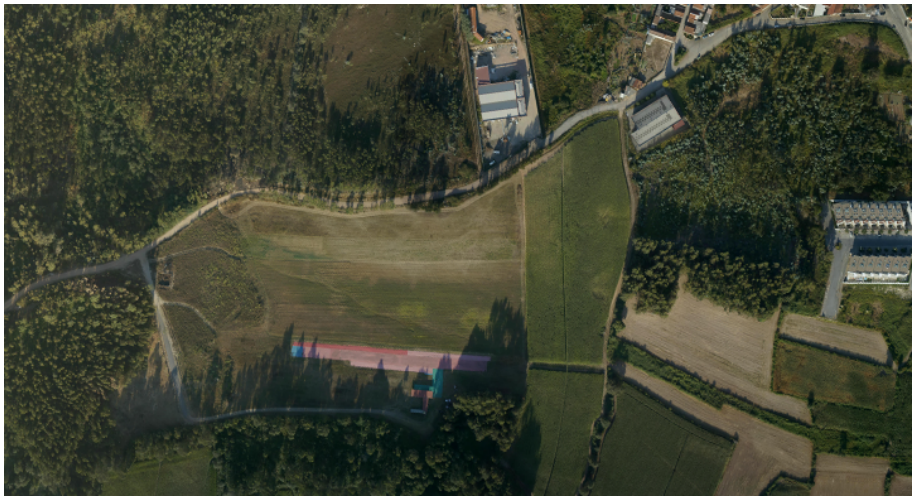


Figure 6.12: LIPA Model Aircraft Airfield (Figure extracted from (lip, 2014)

The tests were conducted in an model aircraft airfield located in *S. Romão do Coronado*. This field belongs to an association by the name of *Liga de Iniciação e Propaganda Aeronáutica* (LIPA) dedicated to model aircrafts with members flying both RC and AutoPilot UAVs.
The airfield is constituted by a 150m long landing strip, on an open space of 2500m^2. However, as we can see in the Figure 6.12, it is surrounded by wooded areas and residential zones, impeding us from flying outside the area depicted in the image.

The two tests were conducted using the same setup in the same location.
The test setup was, using as reference the System described in Section 2.2:

Air Vehicle The X8 platform developed by the LSTS. Two versions of this platform were tested: the X8-01 and the X8-02. As we couldn't test them in the same exact conditions we have no data to compare the UAV's suitability;

Payload: As stated previously, the payload is not a factor we need to take in account for the purposes of this thesis.

Ground Segment: All the tests were conducted on the ground station, a laptop connected via Wi-Fi to the Manta Communications Gateway, which in turn was deploying the 5.0GHz Wi-Fi Network to communicate with the Air Vehicle thanks to a Nanostation M5 radio.

The Manta Communications Gateway was placed in the same location in both Field Tests and we were stationed in the location marked as Ground Station in both tests, as we can see in Figure 6.13:

Figure 6.13: Field Test Location & Setup

6.2.2 Test Plan

Since the opportunity to perform field tests is so scarce, we must maximize their efficiency if we are to draw relevant conclusions. Therefore, in preparation for the field tests, we've created test plans that were to serve as a general guideline regarding what we were trying to conclude in each field test.

6.2.2.1 Field Test #1

Field Test #1 was conducted in an early phase of the program development, without having the Plugin fully developed and all the systems integrated. All the algorithms were fully implemented, so the goal of this test was the validate and test each separate system block. So the test plan was devised as follows:

- Validate the Distance Estimation Algorithm by testing the algorithm for several predefined distances both in the ground and in the air;

- Validate the Position Computation Algorithms by attempting to locate the antenna - placed next to the Manta Communications Gateway, comparing the results to GPS position of the Manta;
- Define more requisites for the final system;

As the Field Test was a mission undertaken by the LSTS to test several other systems, we tagged along in order to withdraw ideas and inspiration to create further additional functionalities and draw conclusions regarding the already existing system blocks already implemented.

6.2.2.2 Field Test #2

Field Test #2 was an entirely dedicated test to the Plugin developed. Here, we tested the program in a much more final state, with several very important tweaks and functionalities added. This test was purely dedicated to testing the project developed for the purposes of this dissertation. Every conclusion drawn in the subsequent sections is extracted from this second test as the results of the first test were underwhelming, at best. Unfortunately, this test was not without its problems. During the first day, everything worked as it was supposed to and we managed to extract several measurements, locating the antenna and estimating the range with great success. But, during one of the take-offs, the UAV used for these tests suffered a small structural failure, preventing it from being used in the following day. So, we used a different UAV, similar to the first one, on the second day. For reasons unknown and beyond our control, the GPS altitude was having erroneous readings on the hardware sensor. This prevented us from doing further tests during the second day. So, our flight test data is very limited but also very promising. We hope that, with continued testing and development this program can easily surpass our objectives and expectations.
The Test Plan for the Field Test #2 was:

- Test and validate the Path-Loss calibration function, by attempting to calibrate the equation in at a pre-defined distance;
- Assess and approve the position acquisition function and its visual representation, by acquiring several positions and validate their distance estimation, visually.;
- Assess the localization algorithms implementation and the accuracy of its position estimation, alongside its visual representation by computing several position estimations, using both two-dimensional algorithms and three-dimensional algorithms;
- Evaluate the Range Estimation function and its visual representation, by observing visually the range estimation drawn and assessing their accuracy and testing their validity using the UAV to navigate the various areas defined by the ranges drawn;

6.2.3 Results

In this section we present the conclusions drawn from the Field Tests done, and assess what could be done to improve this algorithm.

6.2.3.1 Field Test #1

Several problems arose during this week:

HTTP Connection: The firmware used on the Wi-Fi Radio on the aircraft was different from the one used during development. This led to a series of problems connecting to the aircraft to extract the RSSI values. A lot of time was lost fixing this issue, as the implementation had to suffer a major overhaul.

Path-Loss Model: In this stage of development we were using a simplified version of the Path-Loss equation than the one we finally implemented. We concluded that this model was far from optimal, and had a very unsatisfying performance in real world applications.

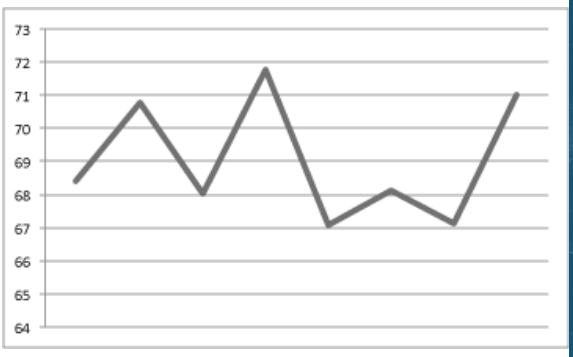
RSSI Smoothing: The factors we initially considered for the smoothing parameters had to be adjusted to provide better results and faster computation.

GPS Error: GPS Error was a bigger hindrance than we first estimated, specially regarding the height parameter. We estimate the error will always be at several tens of meters.

There were many more developments to the program that came from this first week of testing, such as the ability to Save and Load, the Calibration routine, the Manual acquisition mode, and many more small bug fixes and adjustments. It was invaluable to have this first approach to practical situations, even though it was a test with limited success.
Regardless of the difficulties we faced, we managed to successfully detect the antenna thus proving the concept and showing that it was worth the time investment to get the project this far.

6.2.3.2 Field Test #2

First and foremost, in order to achieve a successful antenna location, one must first perform the calibration process. Such routine dictates that we must run perform a flight at a fixed distance from the calibration. Such antenna must be in a known position. Then, measuring the RSSI several times at such a fixed distance we can estimate the gains using equation 5.3 defined in previous sections.
During this calibration test, we ran a loiter maneuver with 75 meters distance to the antenna and measured the RSSI value several times.



(a) Variation of RSSI during calibration

(b) Visual representation in the Neptus environment

Figure 6.14: Calibration Test

As you can see in Figure 6.14a, even performing relatively simple flight maneuvers, the RSSI measure is highly unstable, following no discernible pattern. You can see by the acquisition points - represented by the green dots - in Figure 6.14b that the flight route was relatively stable and yet the results were severely impacted. This variation can be attributed to several factors:

Flight Attitude Although this test aims to minimize such a factor since the aircraft's attitude should remain the same while performing a circular motion, in reality this never happens as the aircraft performs several adjustments that could impact measurements.

Antenna Sensitivity Tolerance Antennas used during this test present a tolerance of $+/-2dB$. We can see in the graphic that the values are for the better part in the $69+/-2dB$ range.

Terrain Topography As you can see in Figure 6.14b, the measurements done south of the antenna have a lower average value than the ones done on the north side. That could be due to a series of reasons, for example obstruction by objects in the signal path.

Antenna Positioning We consider antennas used in this test to be omnidirectional antennas but in practice, they might not be exactly omnidirectional, therefore the ground positioning might have an undesirable effect on the RSSI signal received.

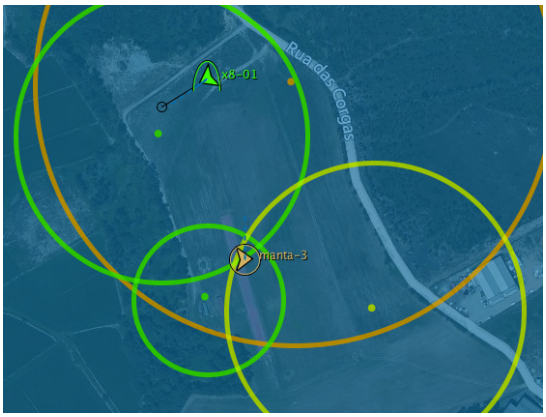
During the field tests we calibrated two UAVs with one antenna, calculating the path-loss constant, shown in the following table. With this constant extracted we can now predict a distance to the antenna from any point desired.

UAV - Antenna Pair	Path-Loss Constant
X8-01 - Nanostation M5	15.5 dBi
X8-02 - Nanostation M5	19.5 dBi

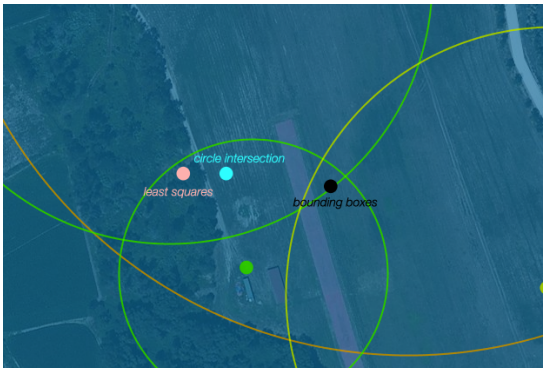
Table 6.2: Path-Loss Constants for different UAV-Antenna Pairs

The difference in these values can be attributed to the differences in configuration of the antennas inside the UAV platform or perhaps, as they were tested and calibrated in two different days, to the weather conditions present.

Locating the antenna is now a matter of using 3 or more points as reference nodes and computing estimated position. In Figure 6.15a we can see four reference nodes chosen and their intersection, clearly in the antenna's position. This is an example of good reference nodes, that will yield an accurate position estimation. With these reference nodes we now must test which position computation algorithms provide better results.



(a) Reference Node Acquisition



(b) Antenna Position Estimation (Offline Test #1)

Figure 6.15: Localization System Test

The results were then compiled in the following table for quick comparison:

Position Computation Algorithm	Error Calculated	Real Error
Bounding Box	3.88 m	8m
Circle Intersection	50.26 m	47m
Least Squares	5944.10 m	70m

Table 6.3: Error Measurements for Different Position Computation Algorithms - Test #1

We can see, first of all that we have a problem with the Least Squares algorithm. It is very susceptible to imprecisions in the point acquisition and since we are using GPS, which is subject to an error, the Least Squares algorithm is very unreliable for estimating the antennas position. Comparing the remaining two algorithms we can see that the **Bounding Box** was the top performer by far, achieving a very acceptable error, although underestimating it in calculation. We performed several more tests and registered the error to draw a conclusion.

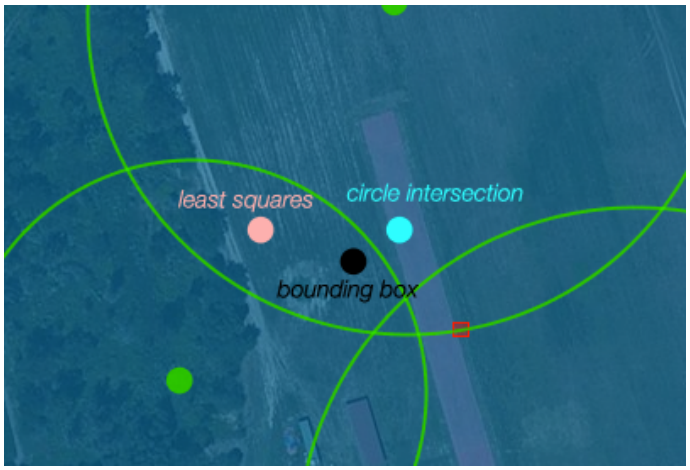


Figure 6.16: Antenna Position Estimation (Offline Test #2)

We performed another Offline Test based on the positions acquired during the field test, seen in Figure 6.16 and verified the real error in relation to the actual antenna position. The results were then compiled into a table similar to the one used previously for algorithm comparison:

Position Computation Algorithm	Error Calculated	Real Error
Bounding Box	0 m	28m
Circle Intersection	11 m	30m
Least Squares	0 m	59m

Table 6.4: Error Measurements for Different Position Computation Algorithms - Test #2

As we can see in the table, the error calculated isn't a good measure to understand real error as it calculates the area of intersection between the several circles but, if the area is small but not centered in the manta (as is the present case), it underestimated the error by a significant margin. We can however say that the localization was achieved successfully, again with a very acceptable error. Here, both the Circle Intersection and Bounding Box perform similarly, both achieving good position estimations for the antenna, represented by the red rectangle. **The Least Squares** algorithm, on the other hand, still presented a significant error margin and we believe it **is not suited for this application**, at least in its current implementation.

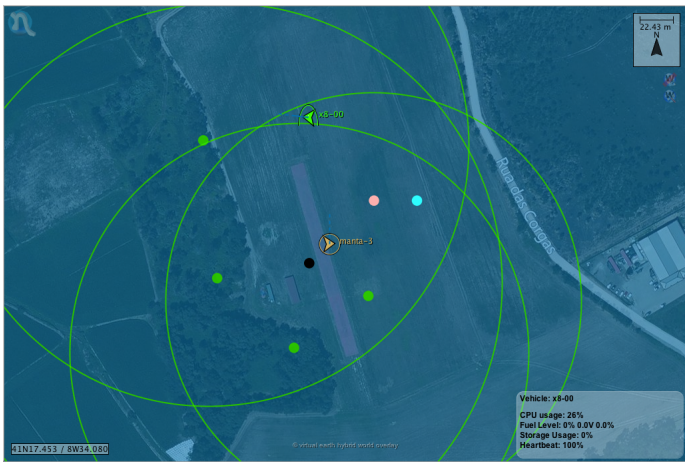
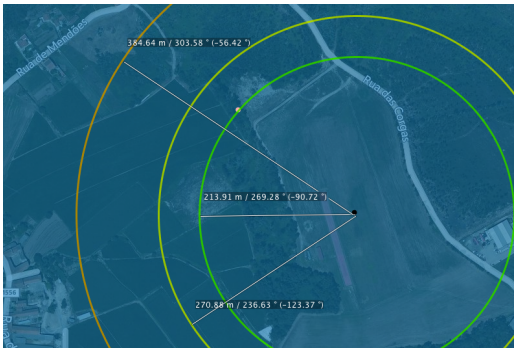


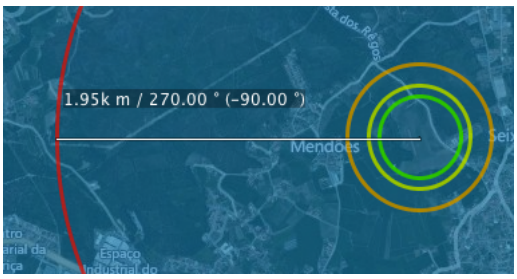
Figure 6.17: Antenna Position Estimation (Online Test)

In Figure 6.17 we can see another localization performed during the field test, in the 9th of January 2014, achieved with success once more. As long as the operator is attentive regarding bad localization estimations (e.g. if a circle overlaps all the others), we can achieve a quality localization every time. Unfortunately, since we only performed a small number of tests and localizations, we cannot provide statistical information regarding the performance of the algorithms. We hope to conduct more tests in further opportunities and have the chance to, not only test the program more extensively but also to put it to good use in practical applications.

Range Estimation was implemented and functional at the date of the second field test. In Figure 6.18a we can see the estimated range, based on a location acquired using the Bounding Box position computation algorithm.



(a) Range Estimation for Various Transmission Rates



(b) Range Estimation for the Minimum Transmission Rate

Figure 6.18: Range Estimation Test

As detailed in Section 5.5.3, the various colors represent different transmission speed capabilities based on the RSSI and are provided by the antenna manufacturer.
The maximum range calculated in this situation was 1.96*km*, as shown in Figure 6.18b.
In theory and according to the signal propagation model we would have connection to the UAV in

a $2km$ radius. We can put in perspective here the insignificance of a $20m$ error when calculating the maximum range of the antenna.
Unfortunately, as our mission area was contained inside the higher transmission rate circle (the green circle), the maximum range could not be subject to test since we did not have safety conditions to perform a test and we did not have equipment that could transmit sufficient enough data to the ground station, we could not test the true purpose of the program. However, we did have experienced operators present in the field test that could confirm that the values were close to the expected and we managed to provide the operators with information regarding the communications network present in our mission area.
Nonetheless, we would require specific field tests to fully test this important component of our system and since this program requires no additional hardware or infrastructures, it can be used and tested at any missions and services provided by the LSTS.

Chapter 7

Conclusions and Future Work

In this chapter, we will evaluate the degree of satisfaction regarding the objectives proposed in the beginning of the dissertation and we will study and propose a series of steps in order to further the development of this work.

7.1 General Remarks

At the start of this dissertation, we set about providing an estimate of the communications range present in a mission area to the UAV operators. The path we chose was a successful one, but not without its obstacles and unexpected turns. We concluded our project with success, achieving the goals we set to complete.
We started development of this system from scratch and it presented itself with a very steep learning curve as it was a multidisciplinary project, that required learning several different subjects in order to fully integrate them together.
We developed a Plugin, *CommRanger*, integrated with the Neptus Command and Control software, capable of locating the Wi-Fi Antennas used to deploy the communications network and estimate their range in the mission area. This program and its functions can now be used as the groundwork in various other studies, such as UAV Wi-Fi Localization Systems, and can be improved up to the point of becoming a tool indispensable to the LSTS and their workflow.
However, in Chapter 6 we encountered some issues and will now revisit them and expand on their magnitude and possible solutions:

7.1.1 Distance Estimation

The estimation of a distance based on RSSI has been widely studied and documented as it is a very convenient means of attaining a distance without the need to resort to additional hardware. Some studies ((Parameswaran et al., 2009),(Wu et al., 2008)) conclude that it is not a reliable parameter for distance estimation. During our studies, however, we found that it can indeed be a usable indicator, but it requires some additional calibrations and/or training in order to be an accurate distance benchmark.

We also concluded, in Section 6.1 that the accuracy of this estimate will be one of the most important, if not the paramount, factor influencing the error of the localization system. As such, improvements done in this section will directly benefit the performance of our program.
Improvements in the filtering section, such as an *Extended Kalman Filter* or *Unscented Kalman Filter* as proposed by Mao et al. could help stabilize the signal and improve our distance estimates.
The location and positioning of the antennas in the UAV are also a concern (see Figure 2.3). Since they are placed in a 90º angle with one facing the top of the aircraft and one pointing towards one of the sides, the antenna coverage will be different on either sides of the UAV. Not only are they placed in a non-symmetric fashion, but they are also placed inside the aircraft fuselage which means increased signal attenuation. Changing the antenna placement could possibly yield much better Distance Estimation results.

7.1.2 Localization System

During the course of this dissertation we tested three position computation algorithms in order to calculate an estimated position of the Wi-Fi antennas used. As we've seen on Chapter 6 of the three algorithms, only two of them performed with acceptable error margins. The Least Squares algorithm faced some severe performance problems and would require a rework by, for example, changing from an unconstrained least squares estimator to a nonconvex constrained weighted least squares estimator, as used in (Cheung et al., 2004). Implementing and testing new algorithms would also be of interest.
Other than that, we achieved localizations with very positive results, achieving extremely acceptable error values: averaging 20 meters.

7.1.3 Range Estimation

As we could not fully test the Range Estimation segment, we cannot attest to its validity. Notwithstanding, we present the user with a range estimation, calculated based on manufacturer provided sensitivity values, representing it visually on the Neptus georeferentiated canvas. The more accurate the RSSI to Distance conversion is, the better that range estimation will be.

7.1.4 Neptus Plugin: CommRanger

The integration of our systems in as a Plugin to the Neptus software was successful and proved to be a very simple way to present the results in a streamlined approach. The representation of the results however requires polish in order to convey the computations more clearly to the operator. The Range Estimation representation for example would benefit by having the different throughput areas represented in a continuous way, changing in color from green to red as the estimated transmission rate diminishes, with transitions representative of the tolerance values inherent to the antennas.

7.2 Future Work

There is wide room for development on the work done in this dissertation as we've noted in previous sections.
We would like to see the Plugin submitted to more thorough tests using several antennas, and analyzed in terms of computational complexity as the number of antennas increase.
There are several other approaches done on Localization Systems that would be interesting to implement and evaluate. Particle Filter Detection and Probabilistic Detection Localization Systems seem particularly interesting and worthy of further investigation.
Using the results from the dissertation and the Plugin developed as basis one could also change the scope of the localization from Wi-Fi Antennas and focus on detecting other UAVs in order to complement the GPS positioning in multi-UAV cooperative flights.
It would also be interesting to see if there are any added computational benefits in running the algorithms on-board the UAV instead of performing all the computations in the ground station.

References

2014. URL http://www.lipa.pt/campovoo/campovoo.html.

2014. URL http://dl.ubnt.com/datasheets/nanostationm/nsm_ds_web.pdf.

Julio Battisti. Redes wireless - parte iii, 2013. URL http://www.juliobattisti.com.br/tutoriais/paulocfarias/redeswireless003.asp.

Jan Blumenthal, Ralf Grossmann, Frank Golatowski, and Dirk Timmermann. Weighted centroid localization in zigbee-based sensor networks. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pages 1–6. IEEE, 2007.

João Borges Sousa, Philip Mcguillivary, João Vicente, Maria Bento, José Morgado, Maria Matos, Granja Pinheiro, Ricardo Bencatel, and Paulo Oliveira. Unmanned aircraft systems for maritime operations.

Azzedine Boukerche. *Algorithms and protocols for wireless sensor networks*, volume 62. Wiley. com, 2008.

Paul Bourke. Intersection of two circles. URL http://paulbourke.net/geometry/circlesphere/.

David W Casbeer, RW Beard, TW McLain, Sai-Ming Li, and Raman K Mehra. Forest fire monitoring with multiple small uavs. In *American Control Conference, 2005. Proceedings of the 2005*, pages 3530–3535. IEEE, 2005.

KW Cheung, Hing-Cheung So, W-K Ma, and Yiu-Tong Chan. Least squares algorithms for time-of-arrival-based mobile location. *Signal Processing, IEEE Transactions on*, 52(4):1121–1130, 2004.

Wan-Young Chung et al. Enhanced rssi-based real-time user location tracking system for indoor and outdoor environments. In *Convergence Information Technology, 2007. International Conference on*, pages 1213–1218. IEEE, 2007.

Jamers R. Clynch. Earth coordinates, 2006.

Academia da Força Aérea. Pitvant - projecto de investigação em veículos aéreos não tripulados, July 2013. URL http://www.academiafa.edu.pt/index.php?bd0b6f49=011.005.004&lang=PT.

Peter De Cauwer, Tim Van Overtveldt, Jeroen Doggen, Filip Van der Schueren, Maarten Weyn, and Jerry Bracke. Study of rss-based localisation methods in wireless sensor networks. In *European Conference on the Use of Modern Information and Communication Technologies (ECUMIT)*, 2010.

Paulo Sousa Dias, João Borges Sousa, and Fernando L Pereira. Networked operations (with neptus). In *3rd annual Maritime Systems and Technologies conference. MAST*, pages 12–14, 2008.

Zhen Fang, Zhan Zhao, Daoqu Geng, Yundong Xuan, Lidong Du, and Xunxue Cui. Rssi variability characterization and calibration method in wireless sensor network. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pages 1532–1537. IEEE, 2010.

Eric W Frew, Cory Dixon, Brian Argrow, and Tim Brown. Radio source localization by a cooperating uav team. 2005.

Daniel L Haulman. Us unmanned aerial vehicles in combat, 1991-2003. Technical report, DTIC Document, 2003.

Ahmet Ibrahim and Dogan Ibrahim. Real-time gps based outdoor wifi localization system with map display. *Advances in Engineering Software*, 41(9):1080–1086, 2010.

Binghao Li, Ishrat J Quader, and Andrew G Dempster. On outdoor positioning with wi-fi. *Journal of Global Positioning Systems*, 7(1):18–26, 2008.

LSTS. Imc documentation, May 2013. URL https://whale.fe.up.pt/imc/doc/master/Navigation.html.

Guoqiang Mao, Sam Drake, and Brian DO Anderson. Design of an extended kalman filter for uav localization. In *Information, Decision and Control, 2007. IDC'07*, pages 224–229. IEEE, 2007.

Ricardo Martins, Paulo Sousa Dias, Eduardo RB Marques, José Pinto, Joao B Sousa, and Fernando L Pereira. Imc: A communication protocol for networked vehicles and sensors. In *Oceans 2009-Europe*, pages 1–6. IEEE, 2009.

William M. Mularie. World geodetic system 1984. Technical report, National Imagery and Mapping Agency, 2000.

W Murphy and Willy Hereman. Determination of a position in three dimensions using trilateration and approximate distances. *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95-07*, 19, 1995.

Ambili Thottam Parameswaran, Mohammad Iftekhar Husain, and Shambhu Upadhyaya. Is rssi a reliable parameter in sensor localization algorithms: An experimental study. In *Field Failure Data Analysis Workshop (F2DA09)*, 2009.

Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. Prentice Hall PTR New Jersey, 1996.

Frank Reichenbach, Alexander Born, Dirk Timmermann, and Ralf Bill. A distributed linear least squares method for precise localization with low complexity in wireless sensor networks. In *Distributed Computing in Sensor Systems*, pages 514–528. Springer, 2006.

J Rullan-Lara, Sergio Salazar, and Rogelio Lozano. Uav real-time location using a wireless sensor network. In *Positioning Navigation and Communication (WPNC), 2011 8th Workshop on*, pages 18–23. IEEE, 2011.

Peter van Blyenburgh. Uavs: an overview. *Air & Space Europe*, 1(5–6):43 – 47, 1999. ISSN 1290-0958. doi: http://dx.doi.org/10.1016/S1290-0958(00)88869-3. URL http://www.sciencedirect.com/science/article/pii/S1290095800888693.

Neeti Wagle and Eric W Frew. A particle filter approach to wifi target localization. In *AIAA Guidance, Navigation, and Control Conference*, pages 2287–2298, 2010.

Xinwei Wang, Ole Bischoff, Rainer Laur, and Steffen Paul. Localization in wireless ad-hoc sensor networks using multilateration with rssi for logistic applications. *Procedia Chemistry*, 1(1): 461–464, 2009.

Rong-Hou Wu, Yang-Han Lee, Hsien-Wei Tseng, Yih-Guang Jan, and Ming-Hsueh Chuang. Study of characteristics of rssi signal. In *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on*, pages 1–3. IEEE, 2008.

Zengbin Zhang, Xia Zhou, Weile Zhang, Yuanyang Zhang, Gang Wang, Ben Y Zhao, and Haitao Zheng. I am the antenna: accurate outdoor ap location using smartphones. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 109–120. ACM, 2011.